2009-06

An activity-driven model for an interactional notion of context

Teo, Hong-Siang

Monterey, California. Naval Postgraduate School

DISSERTATION

AN ACTIVITY-DRIVEN MODEL FOR AN INTERACTIONAL NOTION OF CONTEXT

by

Hong-Siang Teo

June 2009

Dissertation Advisor: Gurminder Singh

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE June 2009	3. REPORT TYPE AND DATES COVERED Dissertation

4. TITLE AND SUBTITLE An Activity-Driven Model for an Interactional Notion of Context	5. FUNDING NUMBERS
6. AUTHORS Hong-Siang Teo	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey CA 93943-5000	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited	12b. DISTRIBUTION CODE

13. ABSTRACT(*maximum 200 words*)

Prior research in context-awareness has largely been dominated by a positivist notion of context. While this notion of context is sufficient for well-defined and focused applications, it suffers from two main shortcomings. First, it fails to consider context as a dynamic construct that arises from a user's interactions. Second, it lacks enough consideration for the role of the human actor in context-awareness. As a result, it is inadequate for dealing with the kind of high-level activities that people naturally engage in as part of their everyday lives. This dissertation proposes an activity-driven model for an interactional notion of context that addresses these shortcomings. In this model, context is defined as a relation between activities. The model was validated using a prototype implementation running on the Google Android mobile phone emulator. Results show that not only does this model improve the computing experience of the user, it also provides unique benefits that have not been available before, such as situation awareness, memory and mental aid, and an associative mode of information access. A rule-based method for discovering parent-child relationships between activities was also validated. These findings demonstrate that the activity-driven model of context warrants further research as a viable basis for context-aware mobile computing.

14. SUBJECT TERMS mobile computing, context awareness, activities as context	15. NUMBER OF PAGES 135
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18 298-102

THIS PAGE INTENTIONALLY LEFT BLANK

AN ACTIVITY-DRIVEN MODEL FOR AN INTERACTIONAL NOTION OF CONTEXT

Hong-Siang Teo
DSO National Laboratories, Singapore
M.Eng (Information Systems Engineering), Imperial College, London, 1997
M.Sc (Electrical Engineering), Naval Postgraduate School, 2006

Submitted in partial fulfillment of the
requirements for the degree of

DOCTOR OF PHILOSOPHY IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
June 2009

Author: Hong-Siang Teo

Approved by: Gurminder Singh
Professor of Computer Science
Dissertation Committee Chair

Geoffrey Xie Professor of Computer Science	Craig Martell Associate Professor of Computer Science
Rudolph Darken Professor of Computer Science	John McEachen Professor of Electrical and Computer Engineering

Approved by: Peter Denning, Chair, Department of Computer Science

Approved by: Douglas Moses, Associate Provost for Academic Affairs

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Prior research in context-awareness has largely been dominated by a positivist notion of context. While this notion of context is sufficient for well-defined and focused applications, it suffers from two main shortcomings. First, it fails to consider context as a dynamic construct that arises from a user's interactions. Second, it lacks enough consideration for the role of the human actor in context-awareness. As a result, it is inadequate for dealing with the kind of high-level activities that people naturally engage in as part of their everyday lives. This dissertation proposes an activity-driven model for an interactional notion of context that addresses these shortcomings. In this model, context is defined as a relation between activities. The model was validated using a prototype implementation running on the Google Android mobile phone emulator. Results show that not only does this model improve the computing experience of the user, it also provides unique benefits that have not been available before, such as situation awareness, memory and mental aid, and an associative mode of information access. A rule-based method for discovering parent-child relationships between activities was also validated. These findings demonstrate that the activity-driven model of context warrants further research as a viable basis for context-aware mobile computing.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. THE MOBILE COMPUTING REVOLUTION

The advent of mobile handheld devices and wireless networking has positioned mobile computing[1] as the next big revolution in computing paradigms. Its disruptive influences can be clearly felt on two fronts. The first relates to productivity. With increased miniaturization, mobile devices are fast becoming as capable and connected in many respects as their desktop counterparts. The result is a trend in the migration of work practices away from the desktop and cubicle-centric environment into more mobile and dynamic settings. A prime example is the evolution of the Blackberry family of mobile devices into the de-facto mobile embodiment of emailing and messaging — a task that, not so long ago, was limited to a desktop environment.

The second disruptive influence of mobile computing is that mobile devices have begun to assume intriguing social roles and identities in our daily lives. A whole slew of social etiquette and practices have evolved around the use of mobile devices, in particular the mobile phones. Separate studies of mobile phone usage in three different cultures — French [1], Scandinavian [2], and American [3] — indicate the extent to which mobile devices have weaved their way into the social fabric of our society. As a society, we are also becoming increasingly attached to our mobile devices in a manner that is inconceivable with our desktop computers. The modern mobile device is as much a statement of fashion and a conduit of social and networking functions as it is a productivity tool.

However, in spite of the tremendous technological and social strides that mobile devices are making, it is indisputable that, as far as harnessing the power of the mobile device and fulfilling the potential of mobile computing is concerned, only the tip of

[1]To avoid ambiguity, the term "mobile computing" in this paper refers to computing with handheld-class devices, which is the main focus of this research.

the iceberg has been explored. To a large extent, this is because of the short sighted approaches that the mobile industry has taken towards mobile computing.

Today's mobile computing typically takes one of two approaches. The first approach can best be described as desktop computing in a mobile device form factor. The idea is to migrate the desktop applications and practices that users are so familiar with onto the mobile device platform. The promise is seamless integration between desktop and mobile realms. It is an attractive, but flawed model. The many ills of desktop computing are well-known [4], and should not be carried over to the mobile domain. Just like desktop computing, this form of mobile computing is piecemeal and application-centric. It tends to force the user to adapt himself to its many design idiosyncrasies and complexities that originate from the desktop world, but are amplified on the mobile device. For example, searching for a particular document would require the user to navigate through the same folder-centric process, but it is much more difficult to perform with limited screen size and input modalities of mobile devices. From the user perspective, this approach is ill-suited for the mobile device platform.

The second form treats the mobile device like a sidekick to the desktop. Popular web services like Google and Yahoo typically have alternate portals for mobile devices. Our own experiences have been that while information retrieval on mobile devices is very functional, information entry is crippled. Users still have to go back to their desktop to manage their information. One example is Google Notebook, a popular Google service that allows the user to store notes and web clippings via its web interface. The mobile web version, however, while allowing the user to view notes, presently does not allow the user to enter notes into any existing catergory except the specific category of "Mobile Notes". Another example is the Microsoft Outlook Mobile Manager. It claims to "brings the power of Microsoft Outlook to your portable device". In practice, it works more like a mobile event notification system, where events are sent from Outlook running on a user's desktop to his mobile device over a choice of delivery modes such as Short Message Service (SMS) or email. Information management via the mobile device is very hard, if not impossible. So while treating mobile devices as a sidekick to the desktop does take into account some characteristics of the mobile device, it does not capitalize on its full power and opportunities.

The upshot is that under current methodologies, getting work done on a mobile device is hard and inefficient. Mobile computing needs to adopt an approach that is more acutely attuned to its particular nature. Certainly, there are untapped aspects of mobile computing that are waiting for research to uncover. One such aspect that has been gathering momentum in the research community is context-awareness.

B. A CASE FOR CONTEXT-AWARENESS

Context is "that which surrounds, and gives meaning to, something else"[2]. Given the dynamic operating environment in which the mobile device is used, context-awareness is clearly one aspect that is of particular relevance to the essence of mobility.

Through context-awareness, one can exploit the dynamic settings that surround the mobile user to deliver services and information that better match the user's needs, intents and goals. While context-awareness does not necessarily depend on mobility, mobility expands the pool of contextual information that can be harnessed to enrich the mobile computing experience. In other words, in mobile computing, the effects of context-awareness can be felt most keenly. A common hypothesis in the research community is that if contextual information can be effectively exploited, many mobile computing tasks can be made significantly easier and more straightforward.

However, existing approaches to mobile computing have not capitalized fully on the uniqueness and strengths of the mobile computing ecosystem, in particular the potential and opportunities afforded by context-awareness, to enhance the user's

[2]Free Online Dictionary of Computing, `http://foldoc.org`. Last accessed 10 Jun 2009.

mobile experience. Conversely, research in context-aware computing, in general, has been slow to adapt to the paradigm shift towards the mobile device. Existing efforts have largely been concentrated on specific forms of awareness, e.g., location awareness and social awareness, and they tend to be application-specific. A general model of context-aware mobile computing has yet to be developed. At the same time, there is generally a lack of a user-centric notion of context that relates more closely to the mobile user's experience, his activities, and his interactions.

C. POSITIVIST VERSUS INTERACTIONAL NOTIONS OF CONTEXT

The notion of context that is predominant in today's context-aware systems has been described as a positivist one [5]. In their seminal paper on the Context Toolkit, Dey et al. [6] presented their definition of context that they claim to encompass prior ideas as follows:

> **Context**: any information that can be used to characterize the situation of entities (i.e., whether a person, place, or object) that are considered relevant to the interaction between a user and an application, including the user and the application themselves. Context is typically the location, identity, and state of people, groups, and computational and physical object.

This is a broad and general definition, and it is representative of the positivist notion of context that is prevalent among current context-aware and ubiquitous systems, mainly because it aligns very well with existing software methodologies and can be easily understood from a bottom-up perspective.

However, context is usually more than the limited view that has been traditionally taken. There are two main shortcomings to this notion of context. First, its failure to consider context as a dynamic construct that arises from a user's interactions, and second, its lack of consideration for the role of the human actor in context-awareness. The upshot is that while this notion of context is sufficient for well-defined scenarios and focused applications, it is inadequate for dealing with the kind of high-level activities that people naturally engage in as part of their everyday lives, such as “write a paper”, “arrange a meeting with team”, “Bob’s birthday” etc. For such scenarios, the contextual set is much more dynamic and unpredictable, and highly dependent on the person’s activity at that time.

As researchers recognize the importance of the human actor in context-aware systems, theories in social sciences and psychology are increasingly being drawn upon to come up with alternate ways of thinking about context that is user-centric, interpretive, and interactional in nature. The goal then is to investigate novel models of context-awareness along this vein that can address the shortcomings inherent in the positivist notion of context.

D. TOWARDS AN ACTIVITY-DRIVEN MODEL OF CONTEXT

With its popularity as a common personal device, the mobile phone is gradually growing into the role of a virtual personal proxy for its user [7]. In doing so, it has begun to encapsulate the many diverse activities that the user engages in. Cypher [8] observed that in reality, “people do not simply perform one activity at a time. Program designers put a great deal of effort into allowing users to perform single activities well, but considerably less effort goes into allowing users to arrange those activities. If computer systems are designed so that they actively support and facilitate multiple activities, they will be more comfortable for the user.”

The same can be said of context-aware systems of today – they do not typically take into account the presence of other activities that may exist in the system, and how they interact and enrich each other contextually. Instead, each application pursues its own application-centric notion of context. For context-aware mobile computing to become a powerful computing paradigm for the future – one that can better match the human experience – mobile systems should take advantage of the opportunities afforded by the knowledge of multiple activities that people engage in while using their mobile devices.

In making sense of an activity-at-hand, we often take into consideration information from our other activities that exist “at the back of our minds”. We often try to place the activity within a larger picture of our other activities. We do this naturally to help us make sense of the current situation.

Consider a simple example in the personal information management (PIM) domain: You receive a message requesting for a meeting. As a result, a calendar event is created for this meeting. Clearly, the message forms part of the context that surrounds the event. It reminds you of why the event was created, and gives meaning to the event insofar as what the meeting is about. But in today’s computing paradigm, both the message and the event are often laid out or accessed in a piecemeal and application-centric way, devoid of the context that exists between them. As a result, the burden falls on the user to maintain that link. If the message and the event is separated far enough in time, or if there are simply too many activities occupying the user’s mind, that link may eventually be lost upon the user. The result is that the user’s ability to make sense of the activities diminishes. On a mobile device, this problem can be exacerbated by its limited user-interface modalities, which forces the user to navigate through multiple layers of screens to find linkages between disparate pieces of information.

Similarly, one of the many things that we can do with our mobile devices nowadays is to bring up a map of our current location. That map can even be annotated with available services in the area. Now, if we can put this information in the context of our various activities that are relevant to that location, such as pending tasks and events, then the location would make more sense to us, because we can rationalize it in the light of our own activities.

So our activities that form a context surrounding that location add meaning to that location. Such a context is dynamic in nature and differs from instance to instance, because our activities change as time progresses. The next time we revisit that location, our activities would have changed, and thus so does the context, and consequently how we treat or use that location.

Here is a model of context that is formed from activities, and where activity and context are unalienable concepts. It is based on an interactional notion of context, versus the positivist notion of context that is predominant in context-aware systems[5]. We call this an activity-driven model of context.

E. MOBILE COMPUTING FOCUS

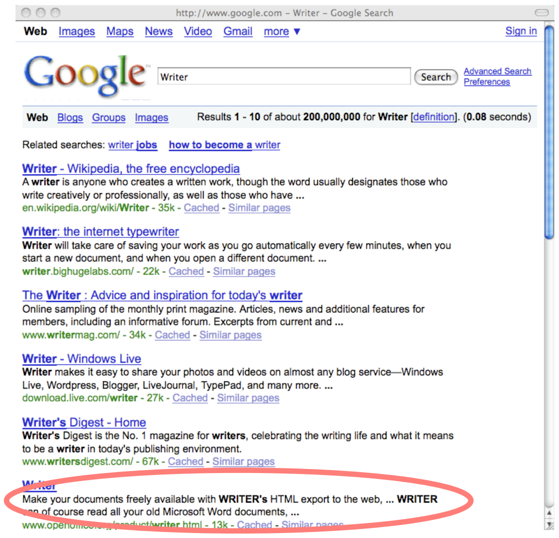
Admittedly, an activity-driven model of context is not exclusive to the mobile computing domain. It can equally be applied to desktop computing for similar benefits too. Consider the following example: Say you are editing a document with OpenOffice Writer[3], and you are unsure about some of its features. So you search the Internet for "Writer" hoping to find some information about it. Using Google as the search engine, you may find that only one out of the first ten results relate to what you want (cf. Figure 1a). However, if the Google search engine could know the context of this search, then by simply appending "OpenOffice" to the search term, the search results would likely be much more appropriate (cf. Figure 1b).

The reason for this disparity in experience is that applications do not have an activity-centric sensitivity to the context in which it is operating. In the end, the burden falls on the human to maintain the links, bridge the gaps, and provide the proper context to each.

Another example can be shown using an application called Fresh[4]. Its main idea is to keep files that one has recently used handy in a "drawer" metaphor, so that there is no need to have to keep searching through the system for them when they are collected into one easy to access location (cf. Figure 2). The idea is certainly inno-

[3]OpenOffice.org, `http://www.openoffice.org`. Last accessed 10 Jun 2009.

[4]Ironic Software, `http://www.ironicsoftware.com/fresh/index.html`. Last accessed 10 Jun 2009.


http://www.google.com – Writer – Google Search
Web Images Maps News Video Gmail more ▼
Sign in
Writer
Search
Advanced Search
Preferences
Web Blogs Groups Images

Results **1** - **10** of about **200,000,000** for **Writer** [definition]. (**0.08** seconds)

Related searches: writer **jobs** **how to become a** writer

Writer - Wikipedia, the free encyclopedia
A **writer** is anyone who creates a written work, though the word usually designates those who write creatively or professionally, as well as those who have ...
en.wikipedia.org/wiki/**Writer** - 35k - Cached - Similar pages

Writer: the internet typewriter
Writer will take care of saving your work as you go automatically every few minutes, when you start a new document, and when you open a different document. ...
writer.bighugelabs.com/ - 22k - Cached - Similar pages

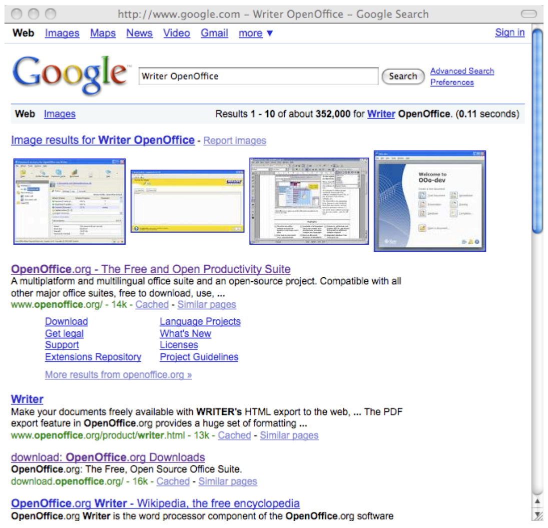
The **Writer** : Advice and inspiration for today's **writer**
Online sampling of the monthly print magazine. Articles, news and additional features for members, including an informative forum. Excerpts from current and ...
www.**writer**mag.com/ - 34k - Cached - Similar pages

Writer - Windows Live
Writer makes it easy to share your photos and videos on almost any blog service—Windows Live, Wordpress, Blogger, LiveJournal, TypePad, and many more. ...
download.live.com/**writer** - 27k - Cached - Similar pages

Writer's Digest - Home
Writer's Digest is the No. 1 magazine for **writers**, celebrating the writing life and what it means to be a **writer** in today's publishing environment.
www.**writers**digest.com/ - 67k - Cached - Similar pages

Writer
Make your documents freely available with **WRITER's** HTML export to the web, ... **WRITER** can of course read all your old Microsoft Word documents, ...
www.openoffice.org/product/**writer**.html - 13k - Cached - Similar pages

(a)


http://www.google.com – Writer OpenOffice – Google Search
Web Images Maps News Video Gmail more ▼
Sign in
Writer OpenOffice
Search
Advanced Search
Preferences
Web Images
Welcome to OOo-dev

Results **1** - **10** of about **352,000** for Writer **OpenOffice**. (**0.11** seconds)

Image results for **Writer OpenOffice** - Report images

OpenOffice.org - The Free and Open Productivity Suite
A multiplatform and multilingual office suite and an open-source project. Compatible with all other major office suites, free to download, use, ...
www.**openoffice**.org/ - 14k - Cached - Similar pages

Download	Language Projects
Get legal	What's New
Support	Licenses
Extensions Repository	Project Guidelines

More results from openoffice.org »

Writer
Make your documents freely available with **WRITER's** HTML export to the web, ... The PDF export feature in **OpenOffice**.org provides a huge set of formatting ...
www.**openoffice**.org/product/**writer**.html - 13k - Cached - Similar pages

download: **OpenOffice**.org Downloads
OpenOffice.org: The Free, Open Source Office Suite.
download.**openoffice**.org/ - 16k - Cached - Similar pages

OpenOffice.org **Writer** - Wikipedia, the free encyclopedia
OpenOffice.org **Writer** is the word processor component of the **OpenOffice**.org software

(b)

Figure 1. Google search results using (a) "Writer" and (b) "Writer OpenOffice".

vative, but it highlights three problems that are symptomatic of current computing paradigms. First, it is file centric, so it does not capture recent activities that cannot be represented by distinct files, such as an event or even an email message (which is typically stored in a monolithic file together with all other email messages). Second, it cannot differentiate between changes made by applications and changes made by the user. For instance, preference files occasionally show up in the list when application settings are modified. Third, and more importantly, the recent file list does not have a relation to what the user is engaged in at the moment. In other words, it is not sensitive to the user's activities. For instance, if one was to resume the writing of a document that was shelved some time back, the list will not reflect say, files that were modified around the time the document was shelved. It will still simply show the most recent list of files as of "now". Again, applications such as Fresh do not have an activity-centric sensitivity of the context in which it is operating.

It is easy to see that if these applications were sensitive to an activity-centric context, the information and services they provide would be so much more useful and relevant. Having said that, the design challenges and expected benefits of the

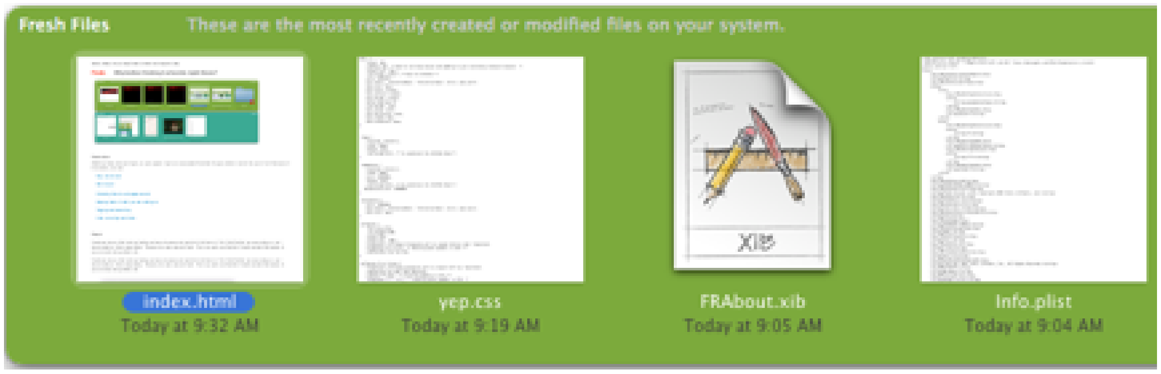


Figure 2. The Fresh application. Source `http://www.ironicsoftware.com/fresh/images/freshdetail.png`.

activity-driven model of context will be most keenly felt on mobile devices. First, the inherent limitations in input-output modalities of mobile devices make dealing with multiple activities and their context an awkward task at best. On the desktop, the difficulty can be alleviated by simply opening multiple windows at once — a luxury that the mobile device do not have. Second, the mobile device has a stronger tie-in with the physical state of the user, such as his location and the people around him. These are useful pieces of information that can be harnessed further to enrich the context model usage. Finally, the mobile device is more likely to capture the kind of high-level, informal, and unstructured everyday activities that the human engages in, which are the type of activities that will benefit the most from this model of context.

The focus of this research is to apply the activity-driven model of context to mobile computing.

F. RESEARCH STATEMENT

This dissertation contends the following:

1. To fully exploit the strengths and potential of the today's mobile devices, there is a need to incorporate context-awareness into the very essence of mobile computing.

2. There is a need to address two shortcomings with the positivist notion of context that is predominant in today's systems: first, its failure to consider context as a dynamic construct that arises from a user's interactions, and second, its lack of considerations for the role of the human actor in context-awareness. The upshot of these shortcomings is that this notion of context is inadequate for dealing with the kind of high-level activities that people naturally engage in as part of their everyday lives.

To addresses these needs, this dissertation proposes an activity-driven model of context as a viable model for context-aware mobile computing that can better match the human experience.

While the principles of activity-driven model of context can equally be applied to desktop computing, it is a better match for mobile computing, where the design challenges and expected benefits will be most keenly felt.

We will develop a prototype implementation running on a mobile phone platform, and conduct a user evaluation of the model. The goal of the evaluation will be to answer the following research questions regarding the model:

1. Does the activity-driven model of context provides utility and improvements to the mobile user for his everyday activities?
2. In what ways is the model useful and better than what is currently available?

G. CONTRIBUTIONS

The primary novel contributions of this dissertation are the following:

- An activity-driven model for an interactional notion of context is described, where context is formed as a relationship between the user's activities, and which arises dynamically from the activity-at-hand. It has been validated through user evaluation that shows that this model not only provides improvement over current mobile computing modes in factors such as information accessibility and efficiency, but also provides new and novel capabilities such as situation awareness and memory and mental aid.

- A method for gathering context between multiple activities on a system is described, where the activity set that comprises context need not be determined a-priori. It makes use of a mediator to allow activities to perform dynamic query and response operations that generate and maintain context, thus enabling context to be a dynamic construct. A prototype implementation of the mediator approach to context gathering on the Google Android mobile phone platform is described.
- A rule-based method for the dynamic discovery of a parent-child relationship between activities.
- For a parent-child relationship context, heuristic for the automatic labeling of newly created activities as a child or sibling is described. These heuristic are grounded on user data.

Secondary contributions of this dissertation are the following:

- Categories of perceived usefulness and improvement have been derived, grounded on user data. These categories can be used as measures to populate survey forms for future studies.
- For labeling activities, it was found that a balanced approach via a combination of manual and automatic techniques, would more likely yield results that make the most difference to the user, instead of a purely automatic or manual approach.

Parts of the research have been published as [9, 10].

H. DISSERTATION OUTLINE

Following this introduction, the dissertation is structured thusly:

- Chapter II presents a detailed review of related literature in context-awareness. Since the body of works is considerable, the focus of the chapter is on related works in notions of context, models and architectures, rather than end user systems.
- Chapter III discusses the notion of activity and the definition of context within the confines of this text. The activity-driven model of context and its core principles will be presented.
- Chapter IV describes the design and implementation effort to translate the principles of the activity-driven contextual model to a prototype implementation on the Google Android mobile phone platform, including a suite of activity classes within the personal information management (PIM) domain.
- Chapter V describes the laboratory setup, experimental procedures and data collection methods used to evaluate the model using the prototype system that was developed.
- Chapter VI presents the methods used and results arising from the data analysis phase. Both quantitative and qualitative analysis and results will be discussed.
- Finally, the dissertation is concluded in Chapter VII. The research questions posed in this chapter will be reviewed, the limitations to the study stated, and recommendations for future works will be made.

II. LITERATURE REVIEW

In this chapter, a systematic review of related works is presented. Since the body of works is considerable, the focus of the chapter is on related works in notions of context, models and architectures, rather than end user systems.

This chapter starts off by reviewing the two theoretical underpinnings of the notion of context, in particular a positivist notion of context versus the an interactional notion of context. We will argue that the latter is a more user-centered approach to context-aware computing. This is followed by a review of how one aspect of user-centered context-awareness has been explored in research, specifically in the use of interaction histories for interpretation and resource management. Then frameworks, toolkits, and mobile platforms that have been proposed in the field of context-awareness will be reviewed. Finally, we touch on related research areas such as activity-based computing and context-mediated social-awareness.

A. TWO VIEWS OF CONTEXT

First, the predominant notion of context taken by today's context-aware systems is reviewed, as well as how such systems interpret context. Following this, some critiques on this notion of context will be presented. Ultimately, the aim is to point out that context is usually more than the limited view that current context-aware systems have taken. It needs to be considered as a dynamic construct, instead of one that is statically scoped. It also needs to take on a user-centric view, rather than an application-centric or system-centric view. Then an alternative notion of context, that is an interactional notion of context, will be discussed where researchers draw upon theories in social science to address the gaps in our current understanding of context.

1. The Positivist Notion of Context

The term *context-aware systems* was first coined by Schilit et al.[11] to describe a new class of applications that are aware of the context in which they are run. They described three important aspects of context as: where you are, who you are with, and what resources are nearby. They further categorized *context-aware computing* into four types: proximate selection, automatic contextual reconfiguration, contextual information and commands, and context-triggered actions. Moran and Dourish [12] referred to context as "the physical and social situation in which computation devices are embedded." Chen and Kotz [13] defined context as the "set of environmental states and settings that either determines an application's behaviour or in which an application event occurs and is interesting to the user." They further differentiate between *active* context that influences the behaviours of an application, and *passive* context that is relevant but not critical to an application.

The common theme that runs through these definitions is that context encompasses the physical, social, and environmental realms. Lucas [14] added a fourth realm — the information realm — that takes into account the "cyber settings" under which mobile computing operates.

In their seminal paper on the Context Toolkit, Dey et al. [6] presented their definition of context that they claim to encompass these prior ideas:

> **Context**: any information that can be used to characterize the situation of entities (i.e., whether a person, place, or object) that are considered relevant to the interaction between a user and an application, including the user and the application themselves. Context is typically the location, identity, and state of people, groups, and computational and physical object.

This is a very general and broad definition, probably with the intent to sufficiently cover the work that has been done on context-based interaction thus far.

However, Winograd [15] commented that “in using such open-ended phrases such as ’any information’ and ’characterize’, it becomes so broad that it also covers everything from the electric power grid or the list of all files on a distant server to the compiler used in creating the application.”

This is the notion of context that is predominant among most current context-aware and ubiquitous systems, mainly because it aligns very well with existing software methodologies. It is representative of “a bottom-up approach to context... The focus is mainly on understanding and handling context that can be sensed automatically in a physical environment and treated as implicit input to positively affect the behaviour of an application.” [6].

An abstract representation of this notion of context is shown in Figure3. It is essentially a sensor-driven model. The application declares a-priori the contextual states that would be useful to the application itself, so that sensor inputs not applicable to these states will likely be ignored by the context system. The application may have a feedback loop to the context subsystem to fine-tune how it mediates between raw sensor input and the contextual states. Other than that, the flow of information tends to be from the bottom up. The Hydrogen context framework [16], for instance, exemplifies this approach. Another useful way of thinking about this notion of context is via a layered or stacked approach, typified by [17, 18], as shown in Figure 4.

Svanaes [19] considers this view of context as a “world view”, where context is a property of the external world, and can be modeled as a flow of information between entities — users, devices and their environment — as they interact. Thus, the emphasis is on acquiring, representing, classifying, and then presenting context in a useful manner to an application. From a social science perspective, Dourish [5] described this view of context as a positivist view, where analysis tends to be objective and quantitative. In this view, context is cast as a representational problem, that is, how context can be encoded and represented. He noted four assumptions that

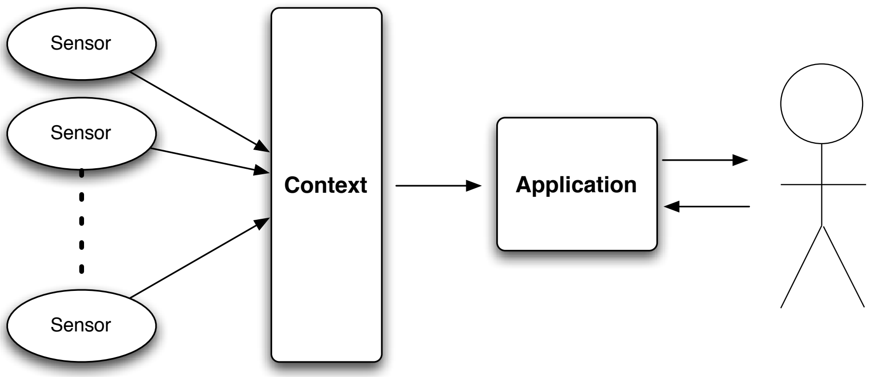


Figure 3. An abstract illustration of the positivist model of context. The arrows indicate the information ows or the interactions between the entities involved. The context system mediates between raw sensor input and the contextual states declared by the application.

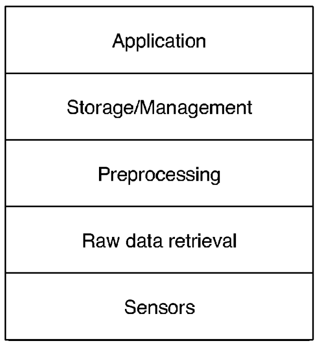


Figure 4. A layered conceptual model for context-aware systems that typifies the positivist notion of context. Source [18].

underlie this notion of context:

(a) Context is a form of information that can be known, coded and represented.

(b) Context can be delineated and scoped in advance.

(c) Context is stable from instance to instance of an activity or event.

(d) Context and activity are separable.

2. Critique

There are two main shortcomings to this way of working with context.

First, it fails to consider context as a dynamic construct that arises from interactions. The determination of contextuality – or of relevance – is not one that can be made a-priori. It is an emergent feature of the interaction, determined in the moment and in the doing [5]. Instead, context should be viewed as a dynamic construct over a period of time, episodes of use, social interactions, internal goals, and local influences. As a result, determining an appropriate set of canonical contextual states in advance may be difficult or impossible. In fact, there are many settings where no canonical sets exists. The appropriateness of a canonical set may also change from moment to moment because internal and external circumstances have changed [20]. In analyzing three well-known theories that are highly relevant to context — situated action[21], activity theory [22, 23], and the locales framework [24, 25, 26] — Greenberg [20] noted that they all suggest that context must be understood as a continually evolving and highly situation-dependent construct. Consequently, any approach that ties the system down to a pre-defined set of contextual states would be sub-optimal on average, or wrong at worse.

Second, it does not sufficiently account for the role of the human actor in context-aware systems. While external factors are relatively easy to capture, internal factors – an individual's interest, history of interaction, current objectives, and the state of the activity he is pursuing – are extremely difficult to acquire [20]. There will also be human aspects of context that cannot be sensed or even inferred by technological means [27]. One can look towards artificial intelligence (AI) to help with the inference, but the state of the art in AI is still quite a ways from meeting these goals. Erickson [28] further pointed out two problems with this approach. First, "piling heuristics upon heuristics in an attempt to map a sparse array of sensor inputs to an actionable interpretation is very different from human awareness. Second, as the set of rules becomes larger and more complex, the system becomes more difficult to understand." The net result is that "our ability to control, and even understand, what is going on around us is diminished."

This is not to say that the current notion of context is flawed, but rather it is incomplete. For many well-defined scenarios, it may well be sufficient, for example telling a cell phone to always vibrate and never beep in a concert if the system can know the location of the cell phone and the concert schedule [12]. Of course, there will always be exceptional conditions, such as when the user is at the boundary of the location. But for the most part, this model works fine for such applications.

The position that is put forward here is that this model will be inadequate for dealing with the kind of high-level activities that people naturally engage in as part of their everyday lives, such as “write a paper”, “arrange a meeting with team”, “Bob’s birthday” etc. For such scenarios, the contextual set is much more dynamic and unpredictable, and highly dependent on the person’s activity at that time. So it becomes more than just a matter of gathering more and more contextual information. More information is not necessarily more helpful. Information is useful only when it can be usefully interpreted, and contextually relevant [12]. Information must be put within the proper context of a person and his activity, which in turn must arise dynamically from his interactions.

In summary, the two main shortcomings of current approaches to handling context are:

1. Failure to consider context as a dynamic construct that arises from a user’s interactions, and
2. Lack of consideration for the role of the human actor in context-awareness.

3. The Interactional Notion of Context

As researchers recognize the importance of the human actor in context-aware systems, theories in social sciences and psychological are increasingly being drawn upon to come up with alternate ways of thinking about context. Coming from the field of phenomenology, Dourish [5] proposed a interactional notion of context, where analysis tends to be subjective and qualitative. In this view, context can be cast as a interactional problem instead of as a representational problem, and the following properties of context holds:

(a) Context is a relational property that holds between objects or activities.
(b) The scope of context is defined dynamically.
(c) Context is an occasioned property that differs from instance to instance.
(d) Context arises from activity. Context is actively produced, maintained and enacted in the course of the activity at hand.

So the central concern with context shifts from the question of "what is context and how can it be encoded?" to the question of "how and why, in the course of their interactions, do people achieve and maintain a mutual understanding of the context for their actions?" In this way, "context becomes an achievement, rather than an observation; an outcome, rather than a premise."

This is a view of context that is centered on a user's activity, hence it represents a user-centric view of context, as opposed to the application-centric or system-centric view of context predominant in today's systems. At the same time, since context arises from activity, context-awareness essentially becomes an activity-driven process instead of a sensor-driven process. However, Dourish did not elaborate on what is the relational property between objects or activities, or how context can arise from activity.

Chalmers [29] also argued that "past social interaction, as well as past use of the heterogeneous mix of media, tools and artifacts that we use in our everyday activity, influence our ongoing interaction with the people and media at hand." Thus, the past can be part of one's context. This concept can be extended to the notion of activity, to argue that not only would activity gives rise to context, an activity can itself become context to other activities or its subactivities.

Chalmers further suggests that there is utility in "making records of the past into useful and practical elements of displays of systems' state and configuration, and tools for system inspection and adaptation, as part of the 'information spaces' used for negotiation and articulation of work and activity." So context should no longer be just some system property peculiar to individual applications and hidden from the user. Instead it should be be meaningfully communicated back to the user, so that he can appreciate the context that surrounds his activity.

B. USE OF HISTORICAL CONTEXT

In addition to Chalmers' use of past interactions with people and systems, and the structures or abstractions over those experiences as resources in people's activity, there are also other research projects that explored the use of a historical context in various ways.

The UMEA (User-Monitoring Environment for Activities) project [30] is a virtual work environment with low-overhead support for setting up and managing personal project contexts. It is informed by activity theory. The system monitors the user's interactions with various document resources, and creates an interaction history mapped to his individual project contexts. By selecting a project the user enters a project context, that is, gets a convenient access to project-related resources. When a new resource is used within a project, the resource is automatically added to the project context.

The system is implemented as an application running under Microsoft Windows, and can run in a foreground mode and a background mode. In the foreground mode, the user can directly view and manage the project resources as well as use a number of personal information management (PIM) tools. In the background mode, the system monitors system events such as opening a web page, printing a file or sending an email, and updates the currently active project with these events. If the event is associated with a new resource, that resource is automatically added to the list of project resources.

There are three main problems with the UMEA system. First, it does not necessary simplify the management of project resources. The project resource list can be cluttered by unrelated events picked up by the system, so it is down to the user to clean it up periodically. Second, the resources need to be ranked in a meaningful way. A document opened and closed several times perhaps need to be shown as more important than a document opened and closed once. Third, switching between project contexts can be awkward, especially for resources that can fall into multiple project contexts.

Along a similar vein, the TimeSpace system [31] provides chronological, activity-oriented workspaces for visualizing and managing information resources within a person's information space. In TimeSpace, an activity is a conceptual grouping of related information items that support particular tasks, projects, or interests. One key feature is that it can be used either alongside or in place of existing hierarchical model of information organization that currently dominates personal computing. When using TimeSpace, the underlying storage structure for these information resources is hidden from the user. They can simply focus on the conceptual and temporal perspectives of the documents as provided by the TimeSpace visualizations.

TimeSpace provides two main interactive visualizations. The first provides an overview of the user's set of activities and operations relating to them over time, with an emphasis on the most recent activities. The second presents a detailed view of the content and development of a selected activity. Access to these information items is through virtual workspaces. A timeline provides a continuous, chronological display of the documents and their changes related to the activity.

In many ways, the concepts of TimeSpace is very similar to UMEA, in that they seek to bring all information items related to an activity under one place, and uses historical interactions with the documents as contextual cues to manage and interpret relationships between documents. Because TimeSpace does not do automatic collection like UMEA does, its workspaces are possibly easier to manage. On the other hand, its information set may not be as rich and dynamic as UMEA's over time.

The MOBIlearn project [32] project aims to explore context-awareness from a user-centered viewpoint, specifically to provide context-aware learning experiences on mobile devices. It considers context as a dynamic process with historical dependencies, that its, context is a set of changing relationships that may be shaped by the history of those relationships. For example, a learner visiting a museum for the second time could have his or her content recommendations influenced by their activities on a previous visit. In this sense, its contextual model and use of historical data is very similar to George Square system [33], where the activities of the person — his locations, photographs taken, web pages loaded etc — are logged and used as a resource to make recommendations to the person on places to go, photographs to take and web pages to load.

C. ACTIVITY-BASED COMPUTING

Conceptually, UMEA and TimeSpace are very close to *activity-based computing*. Design guru Don Norman was one of the first people to attempt to apply human activity theory into a new from of computing — activity-based computing. He described his attempt to realize activity-based computing while he was at Apple Computers [4]:

> The basic idea is simple: Make it possible to have all the material needed for an activity ready at hand, available with little or no mental overhead... [Items] not needed for the current activity are hidden so they do not distract and do not take up valuable work space.

Unfortunately, the effort failed, not because it was a bad idea, but because the mindset of the computer industry was against such a disruptive technology at that time.

Before this effort, the Rooms system [34], released as a product by Xerox, was the first virtual desktop management system which allowed users to organize application windows in different activity spaces — rooms, associated with different tasks — to overcome limitations in screen size. It is not unlike the concept of virtual desktops that was prevalent in Unix and X-Windows systems, and now common in desktop operating systems such as Microsoft Windows and Mac OS X. The Kimura system [35] augments this concept by leveraging on interactive peripheral displays to support the perusal, manipulation and awareness of background activities. Furthermore, each activity is represented by a montage comprised of images from current and past interaction on the desktop. These montages help remind the user of past actions, and serve as a springboard for ambient context-aware reminders and notifications.

More recently, Bardram et al. has implemented an activity-based computing enhancement to the Windows XP desktop operating system [36]. It is a extension of the virtual desktop theme, with a few distinctions. Users can organize applications into individual activity spaces. But unlike a traditional virtual desktop, activity spaces are persistent and stateful across sessions. They can also be saved and resumed. Activity spaces be distributed and migrate across workstations, thus achieving activity mobility and device independence. Results from the research show that users found the enhancement useful for them in their work.

In Bardram's work, we can also find an interesting interpretation and modeling of what an activity is. Bardram [37] defined an activity as "an entity that represents human work activity and contains information about what computational services are supporting this activity". An activity thus comprises:

1. its *purpose* or goal, which currently is just a name and a description,
2. its *participants*, which also includes its owner,
3. its *services* that meet the participants' needs such as applications and data servers, and
4. its *state* such as the state of the services or other stateful data.

In defining activity in this way, an activity becomes essentially a computing resource that is mobile and shareable. It is mobile when the state of the activity can be stored and recalled from persistent storage, and the services are commonly available across platforms, such as web browsers. It is shareable when the different participants can access the services and the data in consistent way. It forms the basis of an activity-based computing framework whose aim is to support real-time, asynchronous and distributed collaboration of human work activities [38].

D. CONTEXT TOOLKITS AND FRAMEWORKS

We have stated that the positivist notion of context aligns very well with existing software methodologies, especially concepts such as object-orientedness and component reusability. So inevitably, several models, frameworks and toolkits have been proposed and implemented whose aims are to support prototyping and developing context-aware systems and applications.

Three organizing models have been proposed for coordinating multiple context processes and components:

- **Widgets.** This model is exemplified by *The Context Toolkit* [39, 6]. Borrowing the idea from graphical user interface (GUI) toolkits, a context widget can be thought of as software abstraction of a hardware sensor. By providing an abstract interface to a class of sensors that provide similar context data, such as a location widget that can draw its data from network sensing or Global Positioning System (GPS), widgets hide low-level details of sensing and simplifies sensor reusability.

- **Networked Services.** This model is exemplified by [40], and resembles the client-server architecture widely used in network settings to provide mobility in information and services access. For example, location information can now be made available via a network service. In fact, there can be several location services with different granularity that the application can choose from. The key advantages to this approach are robustness, high configurability, and, through component discovery services, the potential to be settings independent. The disadvantages are the additional cost and complexity for components to discover and communicate with each other, which are typically higher than the widget approach where the coupling between components is tighter.

- **Blackboards.** Winograd [15] regarded both the widget and networked services approaches to be process-centric. Widgets need to be managed within the process of a widget manager, such as a window manager. Services are typically handled by a process on some processor. In contrast, he proposed a blackboard approach that is data-centric. Rather than sending requests to distributed components and receiving callbacks from them, processes post messages to a shared message board (or blackboard), and then subscribe to receive messages matching a specified pattern that have been posted. The key advantages to this approach are simplicity and flexibility in adding new context sources, since the connection is effectively asymmetric. The downsides, due to its loosely coupled nature, are in lower communication efficiency due to a minimum of two hops required per communication and the use of a general message structure that is not optimized for any particular kind of data or interaction protocol.

The commonality behind all three models is that they are all variations of a toolkit approach, and can be differentiated by the tightness of the coupling between the context-aware application and its components. Greenberg [20] warned of an inherent design trap in this kind of approach: while it is a useful and elegant way to design and implement context-aware applications for simple and routine contextual situations, it does not include anything to inform the designer about what contextual situations are appropriate to it. The result could be uninformed designers building inappropriate or misguided systems. This is a similar theme to what Norman [41] highlighted regarding design errors in reference to the venerable Pinball Construction Set [42]: "It is easy to learn, easy to use, yet powerful. There is no such thing as an illegal operation, there are no error messages — and no need for any. Errors are simply situations where the operation is not what is desired". For a game like Pinball Construction Set, such errors are inconsequential and can even be fun. But in real life situations, such errors may have more severe and unsatisfactory consequences.

A closely related effort to the Context Toolkit is the Java Context Awareness Framework (JCAF) [43]. It is a Java-based programming framework and application programming interface (API) for creating context-aware computer applications. In terms of the architecture and services it provides, it is very similar to the Context Toolkit. The main additions are Java style object-oriented modeling of context information, and the extensive use of pure Java APIs for operations like data serialization, authentication and authorization, and remote method invocation. In a sense, JCAF is an exemplar of using Java to implement context-aware systems. However, it also means that it can only be used on platforms with a supported Java virtual machine (JVM).

E. CONTEXT-AWARE MOBILE PLATFORMS

Most of the early efforts in context-aware systems were not specifically targeted at mobile devices, and typically work under controlled laboratory-like conditions. They also tend to be system-centric in design. Since the coming of age of mobile device, there have been various research efforts in incorporating context-sensitiveness into mobile devices, in particular on the mobile phones. By doing so, they inevitably bring context-aware systems out of the laboratories and into the real world.

Schmidt et al. [44, 45] described an architecture for incorporating context-sensitivity into mobile devices, by embedding phones and PDAs with low level sensors that provides physical and logical cues, such as environmental parameters and host information. When the user's situation, place or activity changes, the functionality of the device adapts to those changes. The bulk of the work is very much on sensing environment cues. While activity was referred to as one of the three axes of context — the other being environment and self — it was not developed in the work. It was also not clear how activity differs from self. Even if activity is used, it looks like it is to be treated as an input to the context system, not unlike other sensorial inputs. Similarly, Gellerson et al. [46] described a multi-sensor approach to context-awareness in mobile devices and artifacts, such as a cup. Their initial mobile phone prototype incorporates light sensors, audio sensors, an accelerometer, a skin conductance sensor, and a temperature sensor, on an external self-contained module linked to the mobile host via a serial line. As a measure of how far the industry has come in terms of mobile device technology, with the exception of the temperature sensor, all the sensors listed above can now be found on the Apple iPhone.

In general, smart phone platforms are very tempting for building context-aware applications because of their computing power, storage, programmability, extensibility, and rich hardware options. Another big factor is that because of inherent constraints in form factor, even a small improvement can lead to a significant impact on the usability of the device.

The ContextPhone [47] is a more recent effort to equip smart phones with software components that allow developers for prototyping context-aware mobile applications on the smart phone platform. ContextPhone runs on off-the-shelf mobile phones using Symbian OS[1] and the Nokia Series 60 Smartphone platform[2]. It consists of four interconnected modules, provided by a set of open source C++ libraries and source code components, that developers can leverage on: *sensors* such as location and phone use, *data communication services* via General Packet Radio Service (GPRS), Bluetooth, Short Message Service (SMS) or Multimedia Messaging Service (MMS), *customizable applications* that can augment built-in applications such as con-

[1]The Symbian Foundation, `http://www.symbian.org`. Last accessed 10 Jun 2009.

[2]S60 Home, `http://www.s60.com`. Last accessed 10 Jun 2009.

tacts list and recent calls list, and *system services* for background launching, error logging and status display. ContextPhone has been used in several research project, including ContextContacts [48, 49].

F. CONTEXT-MEDIATED SOCIAL AWARENESS

There has been some recent research efforts to leverage contextual cues to support other high-level human activities. One such concept is context-mediated social awareness. The AwarePhone[50] is an research project to support mobile collaboration in a distributed setting. The AwarePhone is based on the AWARE architecture, which in turn is based on the Java Context-Awareness Framework (JCAF). The AWARE architecture is a general-purpose software architecture for mediating social awareness in different settings, using JCAF as the underlying engine. The current implementation of the AwarePhone is on the Symbian Series 60 phone platform, and provides a contacts list that shows also shows context cues for each contact, such as personal status set by the user, activity information based on electronic calender, and location information based on proximity to Bluetooth beacons, infra-red (IR) beacons, or cell-based information based on wireless local area networking (WLAN) base stations. Based on these context cues, users can then choose an appropriate way to communicate to another party, be it calling him or her directly, or sending a prioritized message.

Along the same vein, ContextContacts [48, 49] is a project that replaces the built-in address book and recent calls list of the Symbian Series 60 smartphones to show contextual cues about the addressees. Some of these cues are location, time spent in the current location, user-selected alarm profile, whether the phone has been manipulated recently, and nearby people using their phones Bluetooth as the sensing technology. They have conducted a field trial where there is some evidence that these situational cues were useful for social inferences in collaborative spaces. ContextContacts is built on the ContextPhone [47] platform, a prototype platform based around the Symbian Series 60 phone platform for context-aware mobile applications.

While not specific to mobile devices, the Family Intercom [51] is a project whose aim is to explore how contextual information can be used to create a variety of lightweight communication opportunities between co-located and remote family members. Context about the status of the called party is communicated to the caller, so that the appropriate social protocol for continuing a conversation can be performed by the caller. The Context Toolkit was used to design and implement the prototypes.

G. NPS THESES

Two recent theses completed at Naval Postgraduate School (NPS) bear some synergy to the work done in this dissertation, in particular mobile information accessibility.

1. Linking Information for Mobile Use

Myers and Zapata investigated how to group and link information for mobile use in a personal information management (PIM) setting [52]. Their key motivation is to improve a person's ability to use acquired information on a mobile device, as the volume of such information increases rapidly with cheap storage and increased connectivity.

The authors recognized that in information management, the goal is not just to sort or search better, or to make the interface more appealing, but to help the user accomplish some task more efficiently. The approach that they took revolves around the central idea of making a user's information more accessible and to automate the organization of this information once it is placed on the mobile device.

To this end, they implemented a PIM tool called Mobile PIM Master (MPM). MPM facilitates organization and retrieval of information through two automated features. The first is the ability to group related items regardless of item type. The second is the automatic creation of new items based on information extracted from a root item, such as creating and grouping new appointments and contacts when these information are contained in an email about a meeting. Items created in the MPM are compatible with the native applications. This allows users to manipulate all data in a unified interface presented by MPM, rather than in multiple native applications.

MPM allows users to manage related information through the concept of a group. Groups can be manually created by the user, or automatically by MPM based on content analysis. Populating the group can also be done manually or automatically.

MPM demonstrated that it can simplify PIM management tasks such as linking and grouping items for better viewing and retrieval. In that sense, it is largely successful. The concept of a group is akin to a shared context, where the items in the groups are related to each other in some ways. However, the creation, management, and presentation of this context exist outside the normal working processes in the mobile device. For example, when a user reads an email, he probably reads it using his preferred native email application. In that view, how can the grouping or context that he created via MPM be useful, without switching over to the MPM interface, and then having to search for that email again? Furthermore, there is the issue of how items created in the future, such as incoming emails, can be related to existing groups. In the end, as the number of groups grows, the user will still be burdened with the management of these groups, instead of the management of individual files and data types. Within these limitations, MPM feels more like a enhanced folder system, albeit a useful one.

2. Personal Information Search on Mobile Devices

As opposed to Myers and Zapata whose work focused on information management on a single mobile device, Akbas [53] addressed a different problem with information management — information mobility. Today, a person's information can exist on several different physical and cyber locations, such as his mobile phone, his personal digital assistant (PDA), his personal computer, his work computer, the internet etc. Akbas noted that managing information scattered across multiple devices has become a growing problem. This provided the motivation for his work, which is to test the feasibility of providing search ability as part of personal information management for mobility.

Akbas's demonstration implementation involves the integration of Google Desktop Search (GDS) on the personal computer platform, and a custom Java Micro Edition (J2ME) search application on the mobile phone platform. GDS automatically handles searching on the personal computer as well as the internet. The mobile is linked to the personal computer via Bluetooth, so that initiating a search on either end will result in search being performed on both ends. The search results will then be transferred over Bluetooth to be collated at the originating end. Of course, direct connection via Bluetooth relies on proximity, so Akbas's design includes the possibility of remote connection via mobile technologies such as General Packet Radio Service (GPRS), third generation wireless (3G), or wireless local area networking (WLAN) on the mobile phone side.

Akbas's proposal is to use mobile and distributed search methods to manage information distributed in various locations. While there are certainly implementation challenges which he noted, such as the restricted routing of traffic across heterogeneous networks, he did not address a core issue, that is, how this approach compares to other distribution information management approach, such as data synchronization via SyncML[3] which is becoming increasing well supported on multiple mobile and personal computer platforms. If information can be synchronized across the devices, then why is there a need for distributed search?

H. SUMMARY

The key take-away from this literature review is the shortcomings of the traditional notion of context that is dominant in today's context-aware systems, specifically its failure to consider context as a dynamic construct that arises from a user's

[3]Synchronization Markup Language, `http://www.openmobilealliance.org/syncml`. Last accessed 10 Jun 2009.

interactions, and its lack of considerations for the role of the human actor in context-awareness.

To address these gaps in our understanding of context, theories from social science such as phenomenology has been drawn upon to come up with an interactional notion of context, which is more user-centered and emphasizes on a user's interactions as a key resource to shape and interpret context. Already, we are starting to see more research efforts in this direction, such as the use of a person's interaction histories to assist in document and activity management, mobile learning, and tourist guides.

In the next chapter, we will present an activity-driven model for the interactional notion of context, where context is formed as a relation between a user's high-level activities.

III. ACTIVITIES AS CONTEXT

In reviewing the literature on context-awareness in the last chapter, we highlighted two problems with our current understanding of context. First, its failure to consider context as a dynamic construct that arises from a user's interactions, and second, its lack of consideration for the role of the human actor in context-awareness.

Drawing from these considerations, an activity-driven model of context is proposed, where context is formed dynamically as a relationship between activities, and where activity and context are unalienable concepts. The model is built on two core principles: activity-driven context and context interactions, which are based on the need to address these two shortcomings.

We start off the chapter by first clarifying within this text the notion of activity, which is so often taken for granted in context-awareness to simply mean something that people do. Then we define what we mean by context in this text, and present the activity-driven model of context. We will show how the proposed model addresses the shortcomings above.

A. THE NOTION OF AN ACTIVITY

Surveying the literature on context-awareness, the term *activity* has often been mentioned, but has rarely been properly defined within the context of each work. Too often it has been taken for granted to simply mean something that people do. Before an activity-driven model of context can be proposed, it becomes necessary to first clarify what an activity is within the confines of this work.

As a start, consider the definition of activity from the Compact Oxford English Dictionary [54], which is as follows:

1. A condition in which things are happening or being done.
2. Busy or vigorous action or movement.

3. An action taken in pursuit of an objective.
4. A recreational pursuit.
5. The degree to which something displays its characteristic property or behaviour.

Definitions 1 and 3 are probably closer to what is sought here, yet they raise the following questions: what is a "condition"? What are "things" being referred to here? Is activity equivalent to action? How is one activity distinct from another? These are difficult questions to answer without specificity to a situation or a context, and thus contribute to the elusiveness of the definition of activity.

Therefore the goal here is not to arrive at a rigid definition of activity that is all-encompassing or unifying, but rather to arrive at a qualitative description of activity that, whilst it may not be totally water-proof, it should at least be clear and useful to the subsequent understanding of the activity-driven model of context.

Our notion of activities follows very closely from that espoused by *everyday computing* [55, 56], that is, the kind of informal and unstructured activities typically found in our everyday lives.

Just like in everyday computing, activities are continuous in time, a constant ebb and flow of actions and interactions, whose starting or ending point may not always be clear. Familiar examples of such activities are orchestrating tasks, communicating with family and friends, and managing information [55].

Interruption to activities are expected, not just because activities are long running and continuous in time, but also because people naturally engage in multiple activities at the same time. The upshot is that at any one time, an activity can either be current or suspended [57]. Current activities can either be at the foreground under conscious control, or at the background under automatic or subconscious control. An example of concurrent activities is driving a car while talking to someone. (In this case, it is contentious which one is at the foreground and which one is at the background.) Current activities can be suspended, and a suspended activity resumed to be current. The resumption can be opportunistic (e.g., based on availability of information), or as a result of a set reminder.

Drawing from *activity theory* [58, 59, 60, 61], each activity has an objective. The objective is also the motive, which stimulates and excites the person in pursuing the activity. The objective can be a material thing, but it can also be less tangible (like a plan) or totally intangible (like a common idea). It is possible that the objective may undergo changes during the process of an activity [61].

Activities are distinguished from each other according to their objectives. Thus real life situations always involve an intertwined and connected web of activities which can be distinguished according to their objectives [61].

Finally, an activity may be suspended if the objective is perceived to have been reached, or if the user simply lost interest in the objective, or it may morph into another activity because the objective has changed. In some cases, even when an activity has "completed", like a task activity, it may still be contextually relevant to other current and future activities.

We can now summarize our notion of activity as follows:

> An activity is a continuous but interruptible process of actions and interactions that we typically engage in our everyday lives with an objective in mind.

Examples of activities are "do grocery shopping", "plan a meeting", "Junior's soccer game" etc. As an illustration, consider the activity of writing a dissertation. It may be split into separate goals, each embodied by a chapter in the dissertation. Thus in consciously filling in each chapter, we are subconsciously moving towards our objective. Even when the dissertation is submitted, the activity may not die. It may be suspended in our consciousness, much as the dissertation gets archived. A time may come when the activity gets recalled, in which case it will probably acquire a new objective, such as to educate and pass on the experience to another person.

Now that a notion of activity is in place, we can proceed to define what we mean by an activity-driven notion of context, which is a departure from current definitions of context.

The terms used in a discipline often give a clue as to how members "see" that field. Once certain distinctions are accepted and become part of the standard vocabulary, these terms can become a barrier to the reality that lies outside. For this reason, it is a useful exercise to periodically reexamine the language used to express our understanding of the world. — Liam J. Bannon [62]

B. THE NOTION OF ACTIVITIES AS CONTEXT

Building upon the key ideas from the interactional notion of context, in particular [5, 29], we define context as follows:

> **Definition**: The context to an activity is a relational property that the activity holds with other activities, and is that which adds meaning to the activity.

The activity-at-hand is typically the current foreground activity, while the other activities that form a context to the activity-at-hand can be background activities or suspended activities.

This definition augments prior definitions of context with some fresh perspectives. First, it emphasizes an unalienable relationship between activity and context. Specifically, outside of an activity, the term "context" has little meaning. Conversely, referring to context without specificity to an activity is equally dubious. Second, context is situated at the activity-at-hand. As the activity evolves, so does the context. Third, it encompasses the idea of multiple activities as context. Finally, the context formed should add meaning to the activity-at-hand, so it should not be a random or meaningless grouping of activities. Instead the resulting contextual set depends on the type of relationship formed from the activities.

The final point addresses the issue of relevancy. It has been stated that the determination of contextuality – or of relevance – is not one that can be made a-priori. So rather than tying down the scope of the context at the point of design, the contextual set is instead dynamically composed of activities tied together by a relationship, and the relevancy to that relationship is then determined by the activities themselves at the moment of the activity-at-hand. Hence, this definition of context respects the the notion of context as a dynamic construct.

Information traditionally considered to be context, such as location, time, identity etc., can be accounted for by considering them to be properties or states of an activity. For example, an activity such as "do grocery shopping" can have both location and time as properties. We also introduce the concept of a default or null activity that captures the current location, current time etc. of the user, when there are no active activity-at-hand. In this way, context-awareness can be conducted entirely within the framework of activities.

Examples of contextual relationships include but are not limited to the following:

1. Relationships based on objective properties such as past, current or planned activities occurring within say, one day of the activity-at-hand, or activities occurring within close physical proximity to the activity-at-hand.
2. Relationships based on subjective properties such as activities that share common tags with the activity-at-hand, or activities that form a parent-child relationship with the activity-at-hand. These are subjective because they depend on how the user labels his activities.
3. Activities matching an expression specified by the activity-at-hand, such as a range of dates or more generally a pattern like a regular expression.

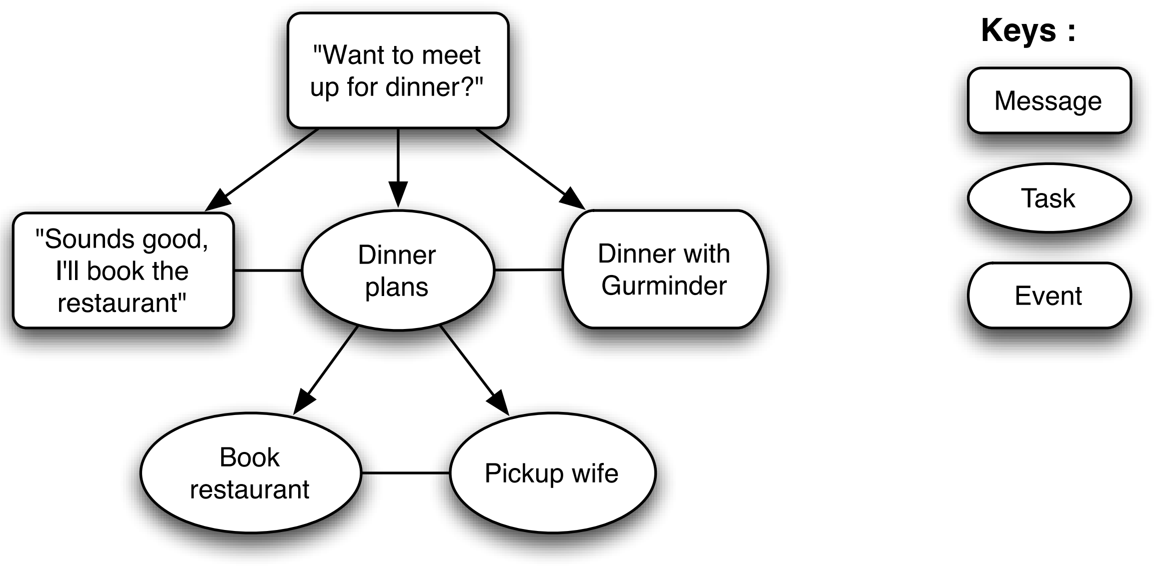


Figure 5. A parent-child relationship context surrounding a task activity called "Dinner plans". The direction of the arrows goes from parent to child. Lines without arrowheads denote sibling relationships.

Figure 5 is a simple illustration of the context of an activity called "Dinner plans". In this case, the contextual relationship is that of a parent-child relationship. The figure shows that the activity originated from an invitation to dinner message received from Gurminder. That triggered other activities such as booking a restaurant (which may in turn involve other activities such as finding a good restaurant, calling the restaurant to make the reservation etc), and picking up the wife (which may involve other activities such as communicating with the wife to decide a pickup time). Then there may be other messages exchanged with Gurminder, and finally the dinner event itself.

With another relationship besides the parent-child relationship, the context arising will have a different look and make-up. Similarly, when centered on another activity, a different contextual picture will emerge. The contextual information will also change dynamically outside and independent of the activity-at-hand as activities evolve. So from instance to instance, activities change due to the user's actions and interactions, and thus so does the context.

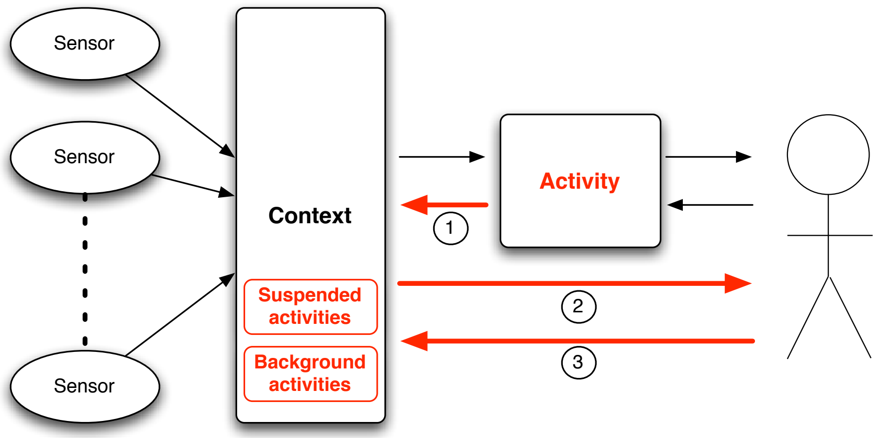


Figure 6. An abstract illustration of an activity-driven model of context. The numbered arrows are the additions to the previous model, and represents the interactions that occur between the human actor, his activity, and context. In addition, the model is now activity-centric, instead of application-centric.

In reality, we create and rely on such relational pictures in our minds all the time to manage our activities. The problem is that we are terrible in recalling and keeping track of them, especially when we are overwhelmed with activities, or when our activities get stretched out over time, thereby stressing our limited short term memory. We may end up simply forgetting to complete a task, or to close a communication loop. Thus, helping the human keep an awareness of the context that surrounds his activity is a natural way of helping him with his activities.

C. AN ACTIVITY-DRIVEN MODEL OF CONTEXT

Our activity-driven model of context is shown in Figure 6. It is intentionally shown here as an overlay over the traditional notion of context (cf. Figure 3) for perspective, as well as to illustrate that this model can work in concert with the traditional approach while still delivering its anticipated benefits to the user.

The model is based on two core principles, *activity-driven context* and *context interactions*.

1. Activity-driven Context

The first principle, activity-driven context, is denoted by arrow numbered 1 in Figure 6. Context is "activity-driven" in two sense. First, the current activity instigates, or drives, the generation of context. This can take place when, for example, the state of the activity changes due to actions taken by the user, e.g., marking a task activity as "done", or when the user switches over from one activity to another. When the new activity becomes current, its context is generated along with it. Second, the resulting context is in turn formed from a relation of activities, in line with the definition of context we have stated. Therefore this principle respects context as a dynamic construct.

2. Context Interactions

The second principle, context interactions, is denoted by arrows numbered 2 and 3 in the figure. Researchers have long argued for context to cease being some system property that is transparent to the user, but should instead be meaningfully communicated to the user for his interpretation. Chalmers [29] pointed out that there is utility in "making records of the past into useful and practical elements of displays of systems' state and configuration, and tools for system inspection and adaptation, as part of the 'information spaces' used for negotiation and articulation of work and activity." In promoting a defensive use of context for context-sensitive telephony, Brown and Randall [63] argued that "context is something which is of great value when it is presented to users themselves to interpret". Even context that might seem unimportant will often be appropriated by users to make sense of their situations. Cypher [8] observed that when two activities are related in context, "the user (a) wants both of the activities to be visible simultaneously; (b) wants to be able to interleave commands to the activities; and (c) wants to be able to pass data back and forth between activities." He termed this phenomenon as *simultaneous interaction* with the two related activities.

So the second principle of context interaction calls for context to at least be

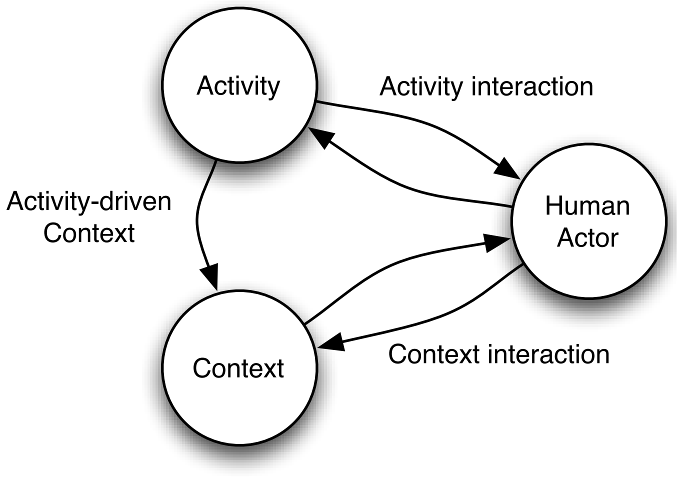


Figure 7. A pictorial summary of the activity-driven model of context. The core principles are activity-driven context and context interactions.

communicated back to the user, so that he may appropriate and apply the contextual knowledge to interpreting his current activity and decides what actions need to be taken. Depending on the implementation, the user may even interact with context directly, such as inspecting relevant information (e.g., the restaurant's location), or switching between contextually related activities via the context.

A pictorial summary of the model is shown in Figure 7. Note the role of the human actor in this model. Also notice that there is a cycle in the interactions between the human actor, his activity, and its context.

One interesting viewpoint arising from this: Imagine in the "Dinner plans" example above (cf. Figure 5), the plans have been finalized. Then you receive a message from the wife that she will need half an hour more before pickup. Upon inspecting the context to the activity, you realize that there is now a conflict with either the restaurant's reservation or the agreed schedule with the other party. Actions now need to be taken so that eventually, the contextual state of the activity can become harmonized again.

So when the state of the activity changes, it may trigger a fresh round of context generation, the result of which is presented back to the user, and the cycle goes on. In other words, reading context and taking appropriate action on an activity becomes a form of harmonization of the overall contextual state of the system that is constantly evolving from both internal state changes as well as external inputs into the system. In Dourish's terms, context thus becomes part of the outcome or achievement of computing, instead of just being an observation or a premise [5], with the user intimately in the loop all the while. Therefore this principle respects the role of the human actor in context-awareness.

D. SUMMARY

The previous chapter highlighted two problems with our current understanding of context. First, its failure to consider context as a dynamic construct that arises from a user's interactions, and second, its lack of considerations for the role of the human actor.

In this chapter, we have described an activity-driven model for an interactional notion of context based on the relationship between multiple activities as context. The model is built on two core principles: activity-driven context and context interactions. We have discussed how the model addressed the two shortcomings above.

In the next chapter, we will describe how a prototype system that embodies the these principles is implemented on a mobile platform. This prototype implementation will subsequently be used as the basis for the evaluation of the model.

IV. DESIGN AND IMPLEMENTATION

In the previous chapter, we have described our activity-driven model for an interactional notion of context. In order to evaluate the model, we need to design and implement a prototype system that embodies its core principles and test it with users. This chapter describes the design and implementation effort.

It starts off by stating the investigation platform and domain. Then we describe how the two core principles of the model — activity-driven context and context interactions — are translated into a prototype implementation on the Google Android mobile platform.

A. INVESTIGATION PLATFORM AND DOMAIN

1. Platform

The investigation of the activity-driven model of context will be conducted on a mobile platform. Granted, the principles of the model are not exclusive to mobile computing, and can be equally applied to desktop computing. However, its effect will be most keenly felt on mobile computing for several reasons.

First, as the mobile device gradually grows into the role of a virtual personal proxy for its user [7], it also begins to encapsulate the many diverse and intertwined activities that the user engages in that are typical of his everyday lives, such as communicating and sharing with friends and family, coordinating and managing tasks, accessing information on the move etc. These are the kind of high-level, informal and unstructured activities that are among the most challenging for context-awareness to address.

Second, the mobile device has a much stronger tie-in with the physical state of the user, for example his location, or the people around him. These are useful and powerful information that can be leveraged on to derive objective contexts for his current activity such as the activities that occur within close proximity, or activities that are shared between the people in a room.

Finally, the inherent limitations in the user-interface modalities of a mobile device can make dealing with multiple activities an awkward experience for the user at best. For example, the mobile user does not have the luxury of opening and viewing multiple activity windows at once, but instead has to navigate through many layers of windows just to locate a specific piece of information that he needs for his current activity. Having such information automatically associated within the context of the current activity would ease the computing experience of the mobile user.

2. Domain

The personal information management (PIM) domain has been chosen as the investigation domain. On most mobile systems, PIM data either already exists, or is easy to generate. Furthermore, there are rich but often overlooked inter-dependencies within the data in the domain that can be exploited via our model of context.

For example, consider task management, which is still a primitive and under-developed concept that has not adequately taken into account the context-sensitiveness of tasks. It is often the case that when a user retrieves a task, the context surrounding that task may be lost upon him with the passage of time — What was the task about? How did it originate? Who else are involved? Moreover, the context surrounding that task may also have changed as a result of changes in circumstances or other activities. Hence, PIM activities like tasks are good candidates for investigating the usefulness of our proposed model of context.

B. ACTIVITY-DRIVEN CONTEXT

The key underpinning of the activity-driven model of context is the notion of context as a relationship between activities. But before one can even take advantage of activities as context, the hurdle of how to gather contextual information from multiple activities has to be crossed.

Gathering contextual information from disparate activities on a system can be a difficult process. The internal state of mobile devices can be as infinitely diverse as their users. This makes it impossible to anticipate a-priori the set of activities that will be present in a mobile device at any given time. Nor do we want to. We have argued that tying down to a pre-defined set activities is sub-optimal on average, and wrong at worse. It also does not differ that much from the positivist approach to context.

What is needed is an approach to context gathering that respects the notion of context as a dynamic construct, that is, being able to harness contextual information from multiple activities in the system without needing to, or when unable to, anticipate the run-time activity state of the mobile system. To this end, a mediator approach to context gathering among multiple activities is proposed.

1. System Overview

The basic idea behind the mediator approach is to make use of a system-level mediator service that is capable of passing messages between activities. Through the mediator service, activities perform dynamic query and response operations that generate and maintain context dynamically. Activities need not be aware of each other's presence; only the mediator needs to be aware of the activities' presence. Conversely, any application that is aware of the mediator and registers with the mediator can partake in the system. Through the mediator, queries from a querying activity is broadcast to other activities in the system. On the other hand, responses and subsequent data exchanges can be directly between activities.

Essentially, the mediator allows for a diverse array of activities to be loosely coupled at run-time. The key benefit of this approach is that it allows contextual information to be gathered from multiple activities in a system where the activity set either cannot be, or does not need to be, determined a-priori.

For maximum utility, the mediator service should be a system-wide service, not unlike popular search services such as the Mac OS X Spotlight service, or the Search-Mananger in the Google Android platform. Both provide access to their respective

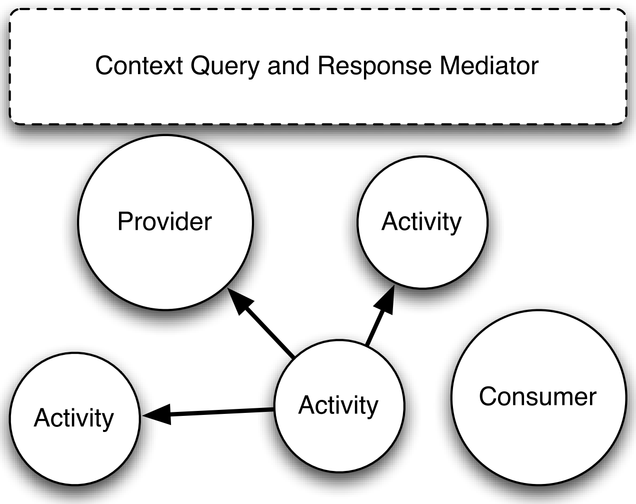


Figure 8. Key players in the mediator approach to context gathering. The arrows illustrate a contextual relationship established by an activity after a query and response operation.

system-wide search services.

2. System Entities

Figure 8 shows the major entities of this system. The system contains objects that can perform one of two roles: context *providers* and *consumers*. Providers are objects that can respond to context queries. They could have interfaces to embedded sensors, or adapters to existing applications that allow their databases to be queried. On the other hand, consumers are objects that only query for contextual information, but do not necessarily respond to queries. An example would be a task list browser that displays the current tasks in the system.

An activity is modeled as an object that encompasses both the provider and consumer role. In other words, not only does it query the system for contextual information, it also responds to context queries from other activities.

3. Context Query and Response

The query and response process works as follows (cf. Figure 9):

- *Query.* Through the mediator service, an activity, typically the current fore-

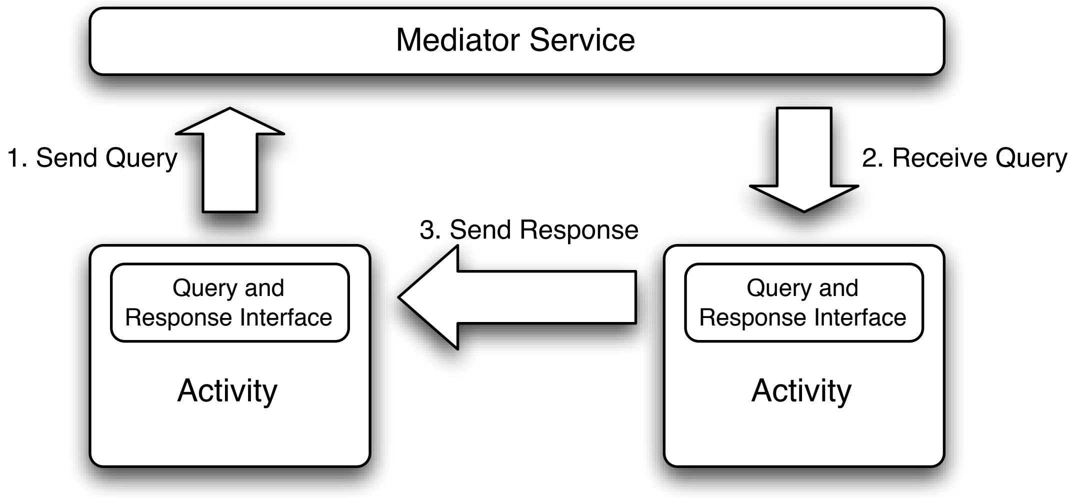


Figure 9. Illustration of the Query and Response process.

ground activity, queries other activities for contextual information. The scope of the query is not determined beforehand. The responsibility of the querying activity is simply to provide a description of the activity and its properties. The more information that is provided in the query, the more likely responses will be received, and hence the richer will be the context that will be subsequently constructed. Conversely, a highly discriminate query can be performed if the information in the query is set to be very specific.

The information supplied in the query is also crucial to determine what kind of contextual relationship results. For example, if a date range is specified, then the resulting context can be a chronology of activities that occur within that specified date range.

- *Response.* An activity receiving a query from the mediator examines the information contained in the query, and determines whether it has contextual relevance to the querying activity. Using the example above, if the query contains a date range, and this activity was active within that date range, then it will respond to the querying activity.

The response can be sent either through the mediator, or directly to the querying activity, with either embedded information, or a pointer to the relevant information. The determination of relevance, and the extent of the information in the response, are entirely within the control of the receiving activity.

Also, an object that has established a contextual relationship with other objects may send asynchronous messages to signal change notifications. This way, fresh contextual information can be maintained at all times.

- *Process.* The querying activity receives the responses either through the mediator or from other activities. From the responses, it constructs a relationship between itself and the activities that responded. In essence, this relation between activities then becomes a context to the activity. The type of relationship formed depends on the information contained in both the query and the responses.

  If the responses are received asynchronously, then the activity has to make a decision on when to stop listening for responses. For example, it can keep listening as long as it is the current foreground activity and then continually update its context, and only stop when it ceases to be at the foreground.

C. IMPLEMENTATION

We translated our system model to an implementation on the Google Android mobile phone platform[1]. The choice of Android over other more mature mobile platforms is based on some unique facilities which Android provide that match well with what we intend to do, in particular *Intents* and *Content Providers.* The former allows context queries to be broadcast to loosely coupled activities, even when they are dormant. The latter allows for contextual data to be shared across activities in a

[1]Google Android, `http://code.google.com/android`. Last accessed 10 Jun 2009.

universal way. This allows the implementation of the query and response process to be kept lightweight.

We adapted the Intents mechanism, so that the Mediator role can be performed by Android itself. Through the application package's manifest file, Android knows which applications can participate in the query and response process. Listing 1 shows the pertinent section of the manifest file of an application that exposes this information to Android.

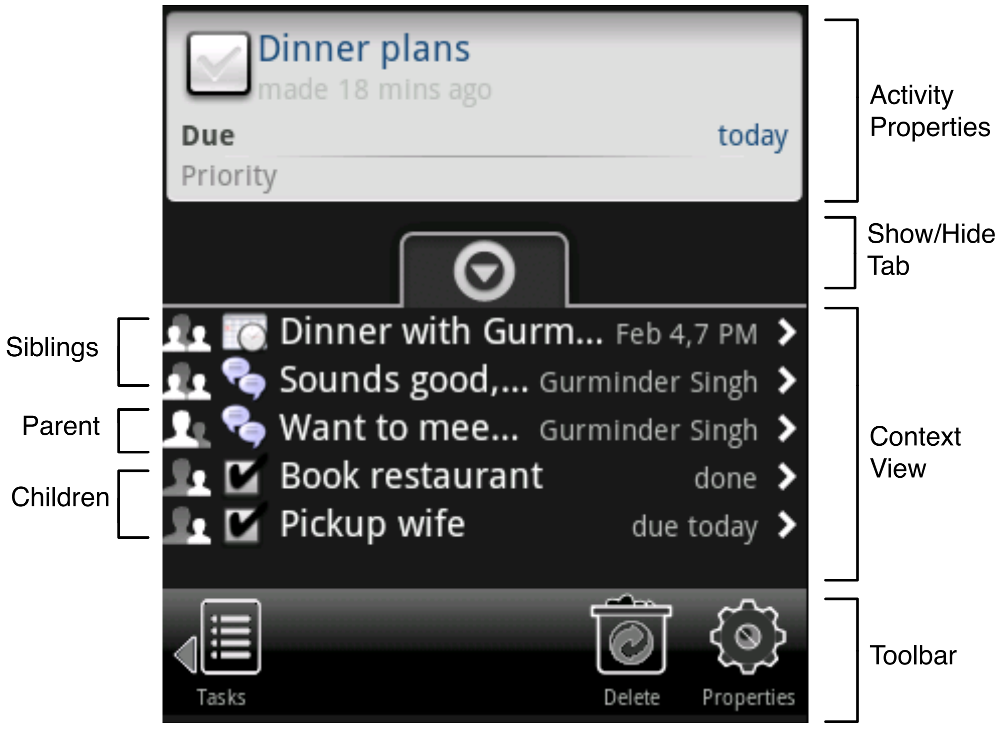
Listing 1 The section of an Android application's manifest XML file that registers the application to participate in the context query and response process, implemented with Android's Intent mechanism. The *receiver* is the Java class that will receive and process the query. The *action* and *category* strings are globals, and is understood and used by all applications for context query.

```
<manifest>
  <application>
    ...
    <receiver android:name=".receivers.ContextQueryReceiver">
      <intent-filter>
        <action android:name="edu.nps.abacas.action.QUERY"/>
        <category android:name="edu.nps.abacas.category.DEFAULT"/>
      </intent-filter>
    </receiver>
    ...
  </application>
</manifest>
```

1. Activity Classes

The investigation domain is the PIM domain. For a start, three activity classes are implemented: Tasks, Events and Messages. Each task, event and message is modeled as an activity.

To a user, an activity screen appears as in Figure 10, which shows a task activity. The top part of the screen shows the essential activity properties. Other properties are available on a separate page. The middle part of the screen shows the context to the activity. In the current implementation, related activities are shown



Dinner plans
made 18 mins ago
Due today
Priority

Dinner with Gurm... Feb 4,7 PM
Sounds good,... Gurminder Singh
Want to mee... Gurminder Singh
Book restaurant done
Pickup wife due today

Figure 10. A task activity screen, showing the task properties, the context view and the toolbar. In the context view, the parent, child and sibling activities are shown.

as a list. The bottom part of the screen contains the toolbar. The “Show/Hide” tab allows the context view to be hidden, so that more of the activity properties can be shown to the user if he so desires.

An event activity screen and a message activity screen would have a similar layout. There may be activity-specific buttons in the toolbar, such as a “Reply” and a “Forward” button for the message activity.

Overall, these classes should be able to provide a rich PIM domain in which to investigate the activity-driven model of context. Furthermore, the model is inherently modular and extensible. So activity classes can be added (or removed) from the system at any time without disruption.

2. Context query

The current activity may query for context whenever it is displayed to the user, or when some of its properties have changed.

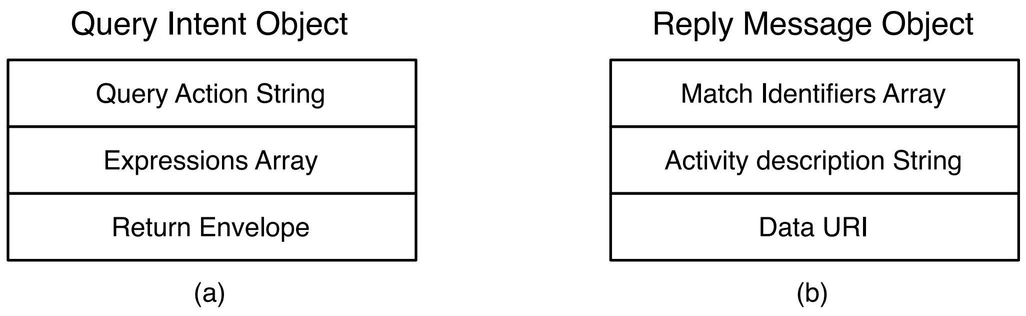


Figure 11. Data structures of (a) context query and (b) context response.

Context queries are conducted using Android's *Intent* broadcast mechanism. An Intent is a passive data structure that holds an abstract description of an action to be performed. Its most common use in the launching of various application functions.

An activity, typically the one that is at the foreground, wishing to query other activities for context first creates an Intent object containing at least three key pieces of information (cf. Figure 11a):

1. The first is a string that identifies a custom action to perform. In this case, this action string is universally recognized by activities in the system as a context query action, and allows the Intent to be directed to activities that can respond to context queries.

2. The second piece of information is a set of *query expressions*. Expression types may include identifiers, dates, times, locations, tags, or general string expressions. These expressions essentially provide a description of the activity, and the type of contextual relationships that the querying activity wishes to form.

3. Finally, a "return envelope" in the form of a *Messenger* object is enclosed in the Intent. The Messenger object allows context producers to be able to send their responses directly to the querying activity.

Once the Intent is populated with these information, it is broadcast to other activities through the Mediator. Currently, the query is implemented using the *send-*

Broadcast system function. This is an asynchronous operation, so the next step for this activity is to wait for responses.

It is important to point out that the querying activity does not know which other activities will eventually receive the query, nor does it know for certain that any response will be sent. Thus, the query is basically a blind query. The scope of the resulting context is therefore one that is not determined by the activity a-priori, but one that is constructed dynamically. This is in line with the objective of treating context as a dynamic construct.

3. Context response

Activities, or more generally context providers, register their ability to respond to queries with Android via their package manifest, as shown in Listing 1. Android then knows how to direct context queries to *Broadcast Receiver* objects defined in the activities.

Within each receiver object, the query expressions and reply Messenger are extracted from the query. Activities inspect the query expressions to determine if there is a *contextual match* with their own data. A match would indicate some kind of relevancy between this activity and the querying activity. The rules and heuristic for the matching are defined entirely by the individual activities, and are likely different from activity class to activity class.

If a match is determined, the activity constructs a *Message* object to carry the response back. The response contains at least three pieces of information (cf. Figure 11b) that helps the querying activity process, present, and access the contextual information returned:

1. The first is a set of constant values that describes the details of the match, for example whether there is a match in identifiers, dates, times, locations, and/or string expressions. This information allows the responses to be sorted and filtered.

2. The second is a set of strings that describe the matching activity in a human readable manner. This is necessary if the contextual information is to be presented to the user, because the querying activity may not know what the matching activity is all about. So it is up to the matching activity to describe itself appropriately so that it can be displayed to the user.

3. The third piece of information is a *data URI* (universal resource identifier) that points to the contextual data. In Android, persistent data is typically stored in SQLite databases owned by individual applications. But through the *Content Provider* mechanism, these data can be shared and accessed across applications. Each record in a database can be addressed via a data URI.

   For example, `content://contacts/people/23` would address a single contacts record (with ID 23) in the contacts database. Thus, by including the data URI in the response, the querying activity now has direct access to the relevant data.

Once the response message object is populated with these information, it is sent back to the querying activity with the Messenger object extracted from the query via the Mediator. As opposed to query, response is a unicast mechanism.

4. Parent-Child Contextual Relationship

Recall that we have defined the context of an activity-at-hand as a relational property that it holds with other activities. Different types of relationships are possible; these have been discussed in Chapter III.

The principal contextual relationship used in the current implementation is the parent-child relationship between activities. As illustrated in Figure 5, our everyday activities typically do have a parent-child structure to them. This contextual information can be a valuable resource for the user as he carries out his activities.

In our current implementation, the parent-child contextual relationship between activities is constructed via the following steps: activity labeling, and relationship matching. Note that while we describe this process within the context of a parent-child relationship, it generally is applicable to other contextual relationships.

Activity Labeling. When a new activity is created from the activity-at-hand, it needs to be labeled as either a child or a sibling to that activity. On the activity screen shown in Figure 12a, the user uses the Menu button to bring up a menu, through which he may create a new activity directly.

The menu presents the user with three options:

1. *New Activity.* The user creates a new activity, and leaves it to the system to automatically label the activity.
2. *New Related Activity.* The user creates a new activity, and labels it as a sibling.
3. *New Sub-Activity.* The user creates a new activity, and labels it as a child.

Upon selecting one of these options, the user will be presented with a pop-up menu (cf. Figure 12b) to select the type of activity to create, i.e., whether it is a new message activity, a new task activity, or a new event activity. After selecting the type of activity, the screen will switch to the new activity.

The system performs the automatic labeling of activities according to a set of heuristic shown in Table 1. At the moment, these are pair-wise relationships between messages, events, and tasks that were derived out of common sense more than anything. In a later chapter, we will examine how well these "best-guesses" hold up against data from users who manually labeled their activities, and whether they have an influence on the results.

Behind the scene, the act of labeling a new activity as a child or sibling sets a key property of the activity.

At the point of creation, each activity is assigned a unique identifier, *ownId*. Each activity also has another property, *parentId*, which is set to the identifier of its parent. The value of this identifier depends on whether the new activity is a child or sibling to the current activity. Listing 2 shows the pseudocode to set the value of this identifier.

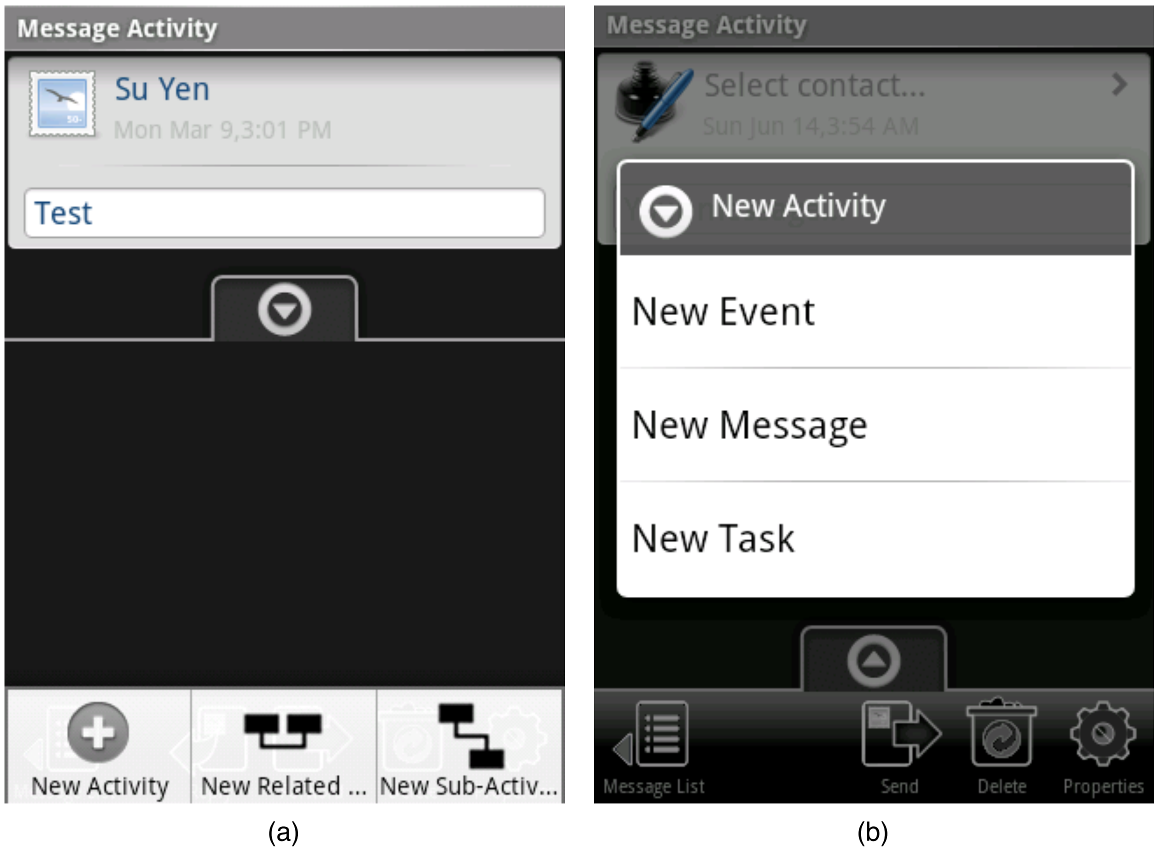


Figure 12. Creating and labeling of new activities. (a) Menu options for labeling a new activity, shown at the bottom of the screen. (b) The type of activity to create.

	Current Activity		
New Activity	**Message**	**Task**	**Event**
Message	child Forward: sibling	child	child
Task	child	child	child
Event	child	sibling	sibling

Table 1. Heuristic for the automatic labeling of new activities.

Listing 2 Pseudocode for setting the *parentId* property of a new activity.

```
set newActivity -> createNewActivity()
if (createAsChild = TRUE) then
  set newActivity.parentId -> currentActivity.ownId
else
  if (currentActivity.parentId = NULL) then
    set currentActivity.parentId -> createUniqueId()
  end
  set newActivity.parentId -> currentActivity.parentId
end
```

Essentially, if the new activity is a child, then its *parentId* will be set to the identifier of the current activity. However, if the new activity is a sibling, then its *parentId* will be set to the *parentId* of the current activity. This is because siblings share a parent. If the current activity does not have a parent, a virtual one is created. In this case, the fact that this *parentId* belongs to a non-existent activity does not affect the relationship building. This is because the relationship is built primarily via a query and response process between activities, and not necessary on static information embedded in the activity.

Thus, when an activity queries for context, both its *ownId* and *parentId* properties will be set as one of the expressions in the query object (cf. Figure 11a) for other activities to inspect for relationship matching, which will be described next.

Relationship Matching. An activity that receives a context query from the mediator inspects the expressions array in the query object. For a parent-child relationship context, the *ownId* and *parentId* properties are inspected, and matched against its own *ownId* and *parentId* properties. From these information, the rules specified in Table 2 are consulted to determine the relationship of the receiving activity to the querying activity.

Except for the match result of *self*, the receiving activity adds the result of the match to the match identifiers array of the reply object (cf. Figure 11b). This

	Querying Activity	
Receiving Activity	**ownId**	**parentId**
ownId	self	child
parentId	parent	sibling

Table 2. Relationship of the receiving activity to the querying activity, based on their *ownId* and *parentId* properties.

allows the querying activity to know whether a child, sibling, or parent has responded. Aggregating the responses from multiple activities, the querying activity builds the parent-child relationship context.

D. CONTEXT INTERACTIONS

In the current implementation, context interactions is provided in the following ways:

Contextual view. The context of the activity is shown to the user together with the essential properties of the activity. In the current implementation, related activities are shown as a list (cf. Figure 10). In effect, this list of activities informs the user of the context to the current activity.

Context-switching. Each activity listed in the context view is an active object, which the user can "click" on. Upon clicking on such an activity, the screen will switch to that activity, and the context view will be refreshed to reflect that of the new activity. In effect, this implements a manner of *context switching* between activities. By doing so, the user is also navigating through the forest of activities through the context view.

E. SUMMARY

We have described the design and implementation of a prototype system on the Google Android mobile platform that implements the principles of the activity-driven model of context. We have stated the investigation domain to be the personal information management (PIM) domain. To this end, activity classes of tasks, messages, and events are implemented. With these tools, we can now proceed to evaluate the model using the parent-child relationship as the principle contextual relationship.

V. EVALUATION

The previous chapters have discussed the activity-driven model of context. Then the implementation of a prototype system on the Google Android mobile phone platform was described, together with a demonstration suite of personal information management (PIM) applications that includes tasks, events and messages. With this implementation in place, the model can now be validated through user evaluation.

This chapter describes the setup and procedure of a laboratory-based user evaluation of the activity-driven model of context, or the ADC model in short.

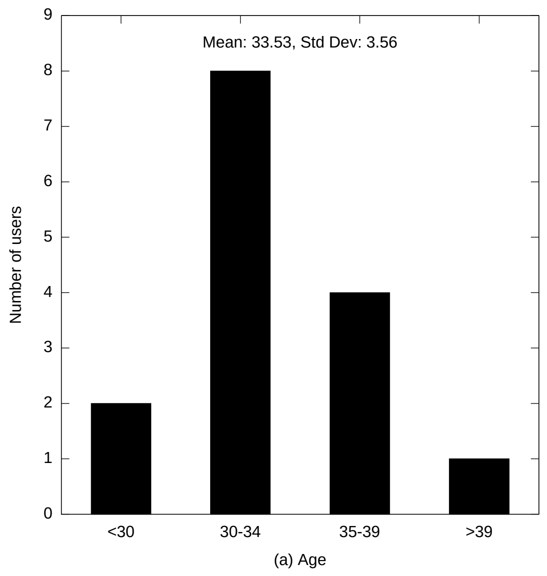
A. OBJECTIVES

The primary objective of this evaluation is to investigate whether and how our ADC model can improve a user's mobile computing experience, and whether it warrants further research as a viable basis for context-aware mobile computing.

To this end, the evaluation aims to answer the following two sets of research questions:

- **Perceived Usefulness.** Quantitatively, to what degree does the user find the ADC model useful to his mobile computing experience? Qualitatively, *why* or in what way is it useful? That is, does it help in making sense of the activity-at-hand? Does the model offer additional interactional capabilities that are valuable to the user's mobile experience?

- **Perceived Improvement.** Quantitatively, to what degree has the ADC model improved the user's mobile computing experience? Qualitatively, *how* has the user's experience improved? That is, does it make it easier or more efficient in interacting with multiple activities? Does it make information more easily accessible? If so, does it expedite task execution?



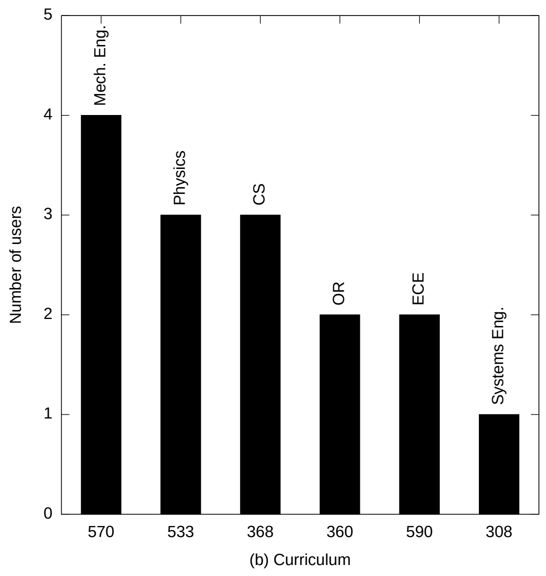


Figure 13. Demographics of users.

Perceived usefulness is chosen as a key metric in this study because it can be seen as “the degree to which a person believes that using a particular system would enhance his or her job performance” [64]. It can also be seen as “a function of task/tool fit” [65]. However, perceived usefulness may not indicate how the system or tool is better or worse than what the person currently has to perform his or her job. This is where perceived improvement fills the gap.

There is clearly a task orientation to perceived usefulness and perceived improvement. Also, both measures contain quantitative as well as qualitative components. These issues will be addressed in the next few sections.

B. TRIAL USERS

A total of 15 postgraduate students from Naval Postgraduate School participated voluntarily for the trial. Figure 13 shows the demographics of the trial user group. Their ages range from 28-41 years, with the mean age at 33.53 years old. 80% of them are in their thirties. Their areas of expertise are well spread out over six different curriculums. While the distribution is not exactly uniform, no one group of

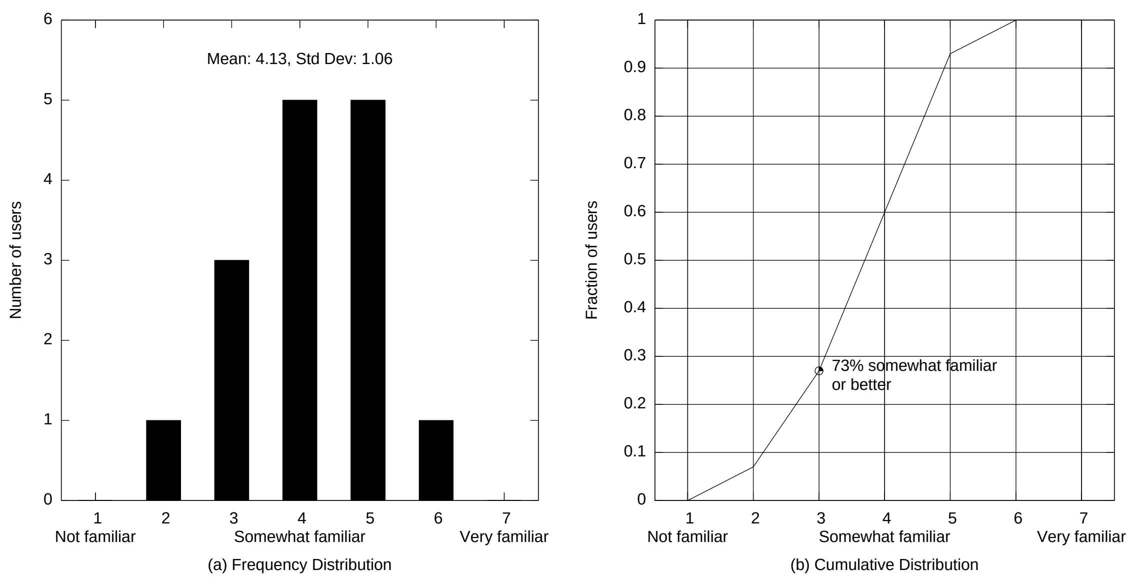


Figure 14. Degree of users' familiarity with mobile technologies.

users dominate the group.

Even so, it is interesting to note that, when asked how they would rate their familiarity with mobile technologies like smartphones and PDAs, 73% of the users responded that they were "somewhat familiar" or better (cf. Figure 14). The average response is slightly better than "somewhat familiar". This is further testament to the degree in which mobile technologies have set foot into our society.

All in all, this user group can be characterized as a young but mature group, who is not unfamiliar with the mobile technology scene, with varied expertise.

C. METHOD

1. Activity Scenario

The scenario that each trial user needs to perform is one of arranging a meeting through the mobile device. To do so, he will execute from an evaluation script (cf. appendix A) which simulates a series of activities involving messages, events, and tasks that typically characterize such a scenario. Each message, event, and task is thus modeled as an activity.

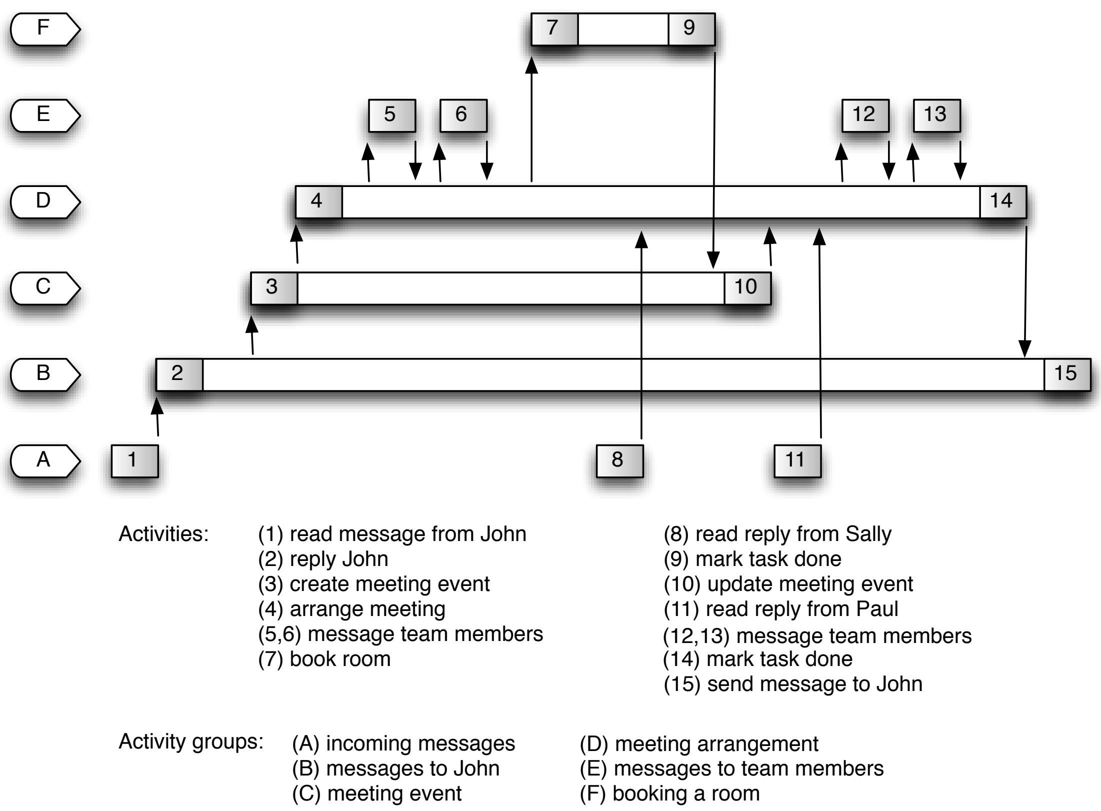


Figure 15. Timeline of the evaluation scenario. There can be up to 15 activities engaged by the user, represented by the shaded boxes. The progression of these activities goes from left to right. These activities have further be logically grouped into six activity groups to help illustrate how the activities are interleaved. The vertical arrows indicate probable transitions between activities.

The scenario starts with an incoming message from possibly a superior of the user, requesting a meeting be set up with the user's project team to discuss some finance matters. Other team members are to be invited. The user then creates events, tasks and message exchanges to effect this meeting arrangement. After the user has judged that all the required tasks has been completed, it ends with a reply to the superior that the meeting has been arranged.

Figure 15 shows a probable timeline of the activities in this scenario. From the figure, it is easy to see that even a seemingly simple task of arranging a meeting can involve a myriad of intertwined activities and interruptions from external inputs that the user has to negotiate. So while this activity scenario is not overly taxing, it suffices to evaluate whether the user can gain benefits from the assist of our activity-driven model of context.

Also, the arrange-a-meeting scenario is chosen because it encompasses three typical and important classes of needs faced by the today's mobile computing users, namely information access, person-to-person communications, and task management. Instead of formulating separate scenarios that focus on each of these needs individually, this study chose to investigate a concerted scenario involving all three that may potentially accentuate the inter-dependencies and nuances between them. However, other scenarios, such as media sharing and multi-party communications, may present other inter-dependencies and nuances that do not present themselves in this case.

2. Procedure

The evaluation process for each trial user proceeds as follows:

(i) Before the trial begins, the user is given an introduction to the purpose of the trial, what he needs to do, and what is expected of him. He is told that the trial will be conducted on a mobile phone emulator. While the emulator is starting up, he is given a brief tour of the user interface elements of the emulator, vis-à-vis the virtual keyboard, the buttons and the application icons.

(ii) Using the mobile phone emulator, the user is asked to perform a series of activities according to a script (cf. appendix A), related to a particular scenario within a specified domain. In this case, the domain is the personal information management (PIM) domain, and the scenario is that of arranging a meeting. The user is encouraged to think aloud during the process.

(iii) The user is required to complete two runs of the script. The two runs are mostly the same, involving interacting with a sequence of messages, events, and tasks, within the larger picture of arranging a meeting. The difference is that in the second run, the user have a view of the parent-child relationship context surrounding the activity-at-hand, and he could also interact with the activities in the context view directly. In effect, the first run establishes a baseline reference that would be typical of today's smartphone platforms, while the second run includes the enhancements to that baseline reference that reflect the support of the ADC model.

(iv) After each run, the user is interviewed. Following [66], each interview is conducted in a semi-structured way according to an interview protocol (cf. appendix B). The audio of the interviews are recorded with the user's knowledge and consent.

(v) During each trial, the researcher sits in with the user for the entire duration of the trial. His main task is to observe and make written notes of the trial process. At the same time, he is also there to offer answers to queries posed by the user regarding any aspects of the trial, although he refrains from helping the user make decisions related to the trial.

3. Control and Flexibility

Even though the user is carrying out his activities prompted by the script, he has both control and flexibility in key aspects of the execution:

- **Control.** The user has total control over the content of the activities that he creates. He is allowed to populate the details of each activities according to his understanding of the scenario, and what he thinks would be best in describing the activities. If the user is to manually label a newly created activity as a child or sibling to the current activity, he is free to choose the relationship according to how he visualizes the overall activity structure.

- **Flexibility.** In the second run, the user has flexibility on how he wishes to interact with activities. While the facility of direct interaction with context is available and made known to the user — that he may create and interact with activities directly from within activities — he is not forced to strictly make use of it at all times. In fact, the user is free to use the original method of interaction if he feels more comfortable doing so. Having said that, at the start of the second run, the user is clearly encouraged to try out the new features. Several users took advantage of this flexibility to ride out the learning curve involved, only starting to interact directly with the context as they move along the script.

4. Laboratory Setup

The Google Android mobile phone emulator is setup to run within a virtual machine instance on a Mac OS X laptop. While the phone emulator can certainly be run directly on Mac OS X, using a virtual machine instance to host the emulator instead allows each user trial to start off from a clean and known state.

This is done easily through the snapshot facility of the virtual machine software. The snapshot captures the known state that the user trial should begin with. After each trial, the slate would be wiped clean for the next user trial by simply reverting to the known snapshot.

D. DATA COLLECTION

User trials were conducted over a three-week period. The length of each trial was between 30-40 minutes. Data was collected from two main sources: interviews and observations.

1. Interviews

Following [66], each user was interviewed in a semi-structured way, according to an interview protocol (cf. appendix B). The interviews lasted between 15 and 30 minutes.

The difference between an unstructured interview, a semi-structured interview, and a fully structured interview is the degree to which the researcher directs the conversational agenda. In an unstructured interview, the researcher would simply suggest a topic, and the interviewee would be free to express whatever opinions he has in any way he wishes. Both the exploration of the topic and the answers given are entirely open ended.

The semi-structured interview, like the one used in this study, would guide the discussion through specific questions designed to explore the designated topic in a certain way, so as to seek out specific information regarding that topic. That is the purpose of the interview protocol — it serves as a guideline for the conduct of the interview. Depending on the user's response, not all questions may be asked, and some questions have duplicate purposes. The main goal during the interview is to prompt the user to express his views and perceptions of what he has just experienced or observed in his own words. As a guideline, the interview protocol ensures that the interviews do not veer too far off on a tangent. But the answers given are still very much open-ended as far as the substance is concerned.

At the other end of the spectrum, the fully structured interviews would basically resemble a survey, where the questions are set in stone, and the interviewee's response is limited to the format of the interview.

The interview is underpinned by four main questions which generate the primary data that will be used in the analysis phase:

- Question 3 collects data relating to the user's familiarity with mobile technologies like smartphones and PDAs.
- Question 4 collects data relating to the user's perception of how the basic version of the tools used in the first run are similar or dissimilar to what they use on their own smartphones.
- Question 10 collects data relating to the user's perception of how useful our ADC model is in helping him carry out his task.
- Questions 15 collects data relating to the user's perception of how much improvement has our ADC model brought to his normal experience.

These questions generates both qualitative and quantitative data. To do so, they are formulated as questionnaire-style questions asked in a verbal way. For example, question 4 would be asked verbally as follows:

> *On a scale of 1 to 7, 1 being not similar, 4 being neutral, and 7 being similar, how do you think the applications you have just used are similar or dissimilar to the mobile PIM applications that you have used?*

The main advantage of this verbal approach, over the more conventional post-event pen-and-paper questionnaire approach, is the ability to directly and immediately probe the user after each main question for details while his thought processes are still fresh in his mind. For example, after the previous question the user may be probed:

> *In what way was it similar/dissimilar?*

Another advantage of this approach is for the user to be able to give fuzzy ratings, like saying "between 6 and 7". For analysis purposes, that particular response will be translated to 6.5, which is probably a truer reflection of the user's intent, instead of forcing him to choose either 6 or 7.

Besides the main questions, there are also setup questions and probing questions. The setup questions prepare the user mentally for some of the key questions. These include questions 5 and 6, and questions 8 and 9. Through questions 5 and 6, the user recounts his experience in the first run. By doing do, the user is able to rationalize his experience on the relatively new smartphone platform against his experience on his own ssmartphone, so that he can gain some points of reference before the second run. Through question 8 and 9, the user recounts his experience in the second run, which allows him to prepare his thoughts for key questions 10 and 15. Setup questions also has a role to play in mitigating the risk of false sense of improvements.

On the other hand, probing questions aim to dig deeper into the user's thoughts for additional insights relating to his trial experience. These include questions 7, 11, 12, 14, and 17.

Finally, the interview closes with a broad, general question. This question gives the opportunity for the user to put his trial experience into a larger context, and helping to remove any tunnel vision that he may have developed during this exercise. By doing so, it is hoped that the users may be able to give additional insights regarding their trial experiences, or any other related experiences.

2. Observations

During each trial, a researcher sits in with the user in a semi-passive observer role. While the researcher would answer queries posed to him by the user regarding aspects of the trial and functionalities of the tools, he does not actively take part in the decisions and actions of the user.

Instead, through audio and visual observations, the researcher notes down interesting actions and thoughts expressed by the user during the trial. In particular, the researcher documents the activity structure that results from the way he labels his newly created activities.

E. SUMMARY

In this chapter, the method used to evaluate the ADC model through user evaluation was described. A total of 15 post-graduate students voluntarily took part in the evaluation. The evaluation was based on a scenario of arranging a meeting on the mobile device. The experimental procedure, laboratory setup, and data collection methods were described in details.

Following the procedure described above, user trials were conducted over a three-week period. After this, the audio of the interviews were fully transcribed, and collated with other data such as data from the researchers observations. Both qualitative and quantitative data were collected. In the next chapter, the analysis of the data and results arising will be discussed.

THIS PAGE INTENTIONALLY LEFT BLANK

VI. ANALYSIS AND RESULTS

This chapter presents the methods and results arising from the analysis of the data collected from the evaluation of the activity-driven model of context, or the ADC model in short.

It starts off with an analysis of the perceived usefulness of the ADC model, then followed by an analysis of the perceived improvements. Each of these analysis include both quantitative analysis using statistical and graphical methods, as well as qualitative analysis using the method of open coding applied to the interview data. Therefore, the results are not just clear indications of the positive or negative utility of ADC model, but also an understanding of why it works or why it does not work from the users' point of view.

Further, an analysis of the pros and cons of automatic labeling versus manual labeling of newly created activities will be presented.

A. PERCEIVED USEFULNESS

The goal of the analysis here is to find out if users perceive the ADC model to be useful to carry out their activities. For example, does it help in making sense of the activity-at-hand? Does it offer additional capabilities that are valuable to the user's mobile experience?

1. Quantitative Analysis

In general, users found the ADC model to be useful to their computing experience (cf Figure 16). On a scale of 1 to 7 — 1 being not useful, 4 being somewhat useful, and 7 being very useful — users gave an above-average score of 5.13.

The standard deviation of 1.68 indicates some degree of variability in users' opinions. This is not entirely unexpected, given that this is the first time they have used something like this, and the limited exposure to it. Nevertheless, 80% of the

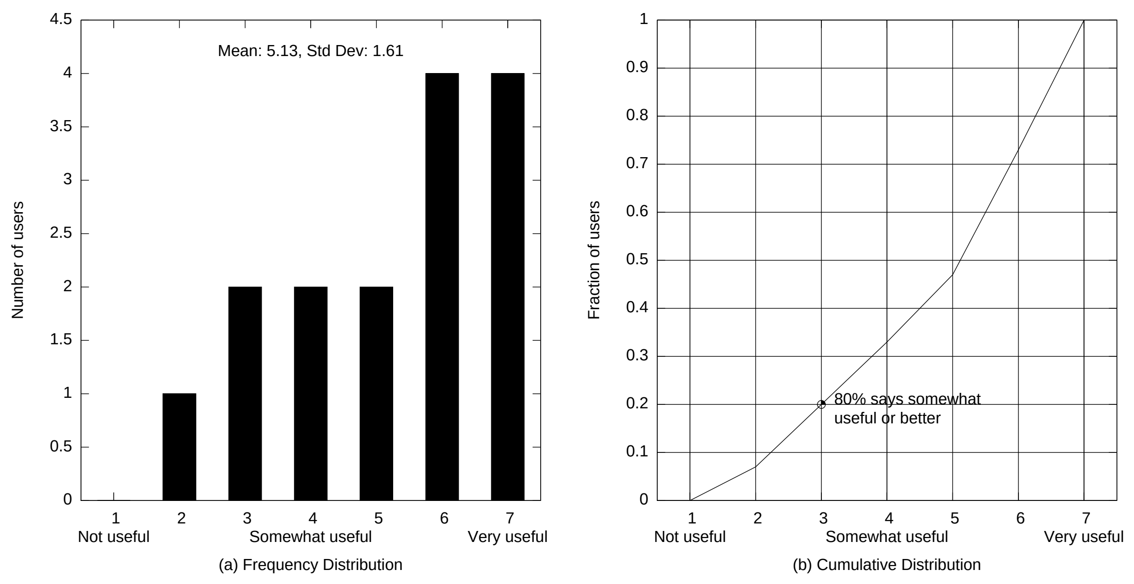


Figure 16. Perceived usefulness of activities as context.

users felt that it was “somewhat useful” or better, with more than half (53%) of the users giving it a high score of 6 or 7. So despite some apparent variability in the data, users’ response was clearly in the positive.

2. Open Coding Analysis

In order to find out why majority of the users found utility in the ADC model, the qualitative analysis method of open coding [66] is applied to the data gathered from the interviews conducted.

Open coding can be done in several ways with varying granularity. The first is line-by-line analysis, which involves close scrutiny of each phrase and even each word in the data. The second is analysis at the sentence or paragraph level. The third is analysis at the document level. In this case, open coding is performed at the sentence level.

Figure 17 shows how the open coding method is employed on an actual paragraph of interview data. In this example, three categories of usefulness is derived. This particular user found activities as context useful in organizing activities and interpreting information. However, the user interface was inadequate in showing the

Data	Categories
“[Very] useful, because it helps me sub-task what are the things to do for the particular main task[**organizing activities**]. It also tells me which one is the parent, which one is the child in a directory format[**interpreting information**]. But how does the task link to the event, that one can be improved further[**inadequate user interface**].”	“organizing activities” “interpreting information” “inadequate user interface”

Figure 17. An example of applying open coding at the sentence level to interview data to derive categories of usefulness.

relationships between activities.

3. Categories of Usefulness

Using this technique, a set of categories of usefulness is derived. The aggregated result is shown in Figure 18. Each category has a popularity score, which reflects the number of users whose remarks fall under that category.

It is important to emphasize here that these categories were not pre-defined by the researcher for the user to choose from. Instead, they were entirely extracted from the interview data using the open coding technique described above. Since the interviews were conducted in a semi-structured way, the answers given were very much open-ended. So the categories described as follows are reflections of perceived usefulness from the users' point of view, subject to a degree of interpretation in terms of classification on the part of the researcher.

Majority of the users felt that the key benefit of the ADC model is in providing contextual information that helped them gain better *situational awareness* of what is going on surrounding the activity-at-hand. There are two aspects to this.

The first is in helping to keep track of the status of things. It gives the users a better feel of how things have progressed, and what things are still outstanding, including milestones that has been set that need to be accomplished, or subtasks that need to be completed before things can move along. In this way, one can track the status of activities as they progress.

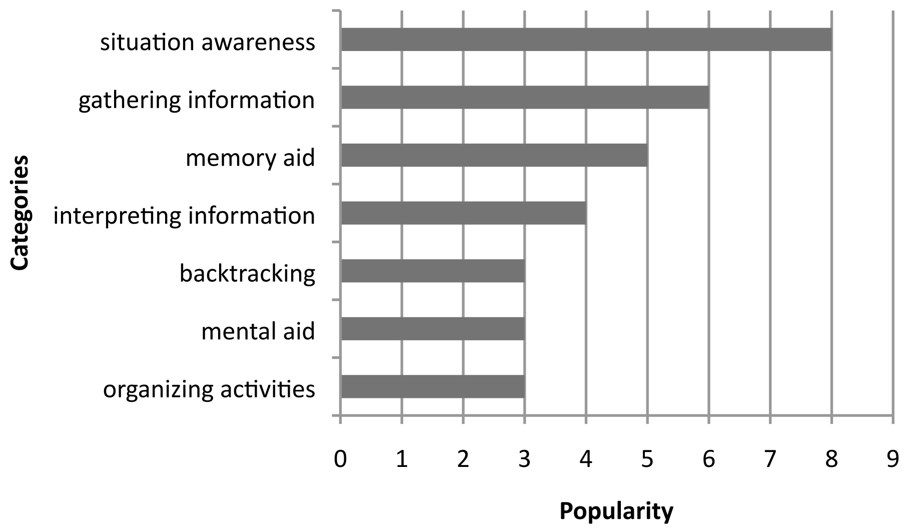


Figure 18. Categories of Usefulness. Each category has a popularity score, which reflects the number of users whose remarks fall under that category.

The second, and perhaps more interesting and powerful, aspect of better situational awareness is that users appeared to have gained an additional perspective into their activities. One user commented how in terms of a time line, he is able to know where he is right now.

> *Grouping all relevant activities together, so that I can track what are the activities that I need to do, and what are the activities that I have done. In terms of time line, it states where I am right now.*

Other users similarly noted that they were able to know what activities have happened in the past that is related to that activity, kept together within one glance and in one space. This perspective of letting users know where they are at that moment of the activity-at-hand in the context of other activities can be akin to having an *activity trail*.

In the evaluation scenario used in this study, the activity trail would comprise the various messages exchanged between the user and his team members, the events, and the tasks. But due to user interface limitations in the current implementation, the user would not be able to see the activity trail in its entirety. Only a snapshot of it centered on the activity-at-hand is shown. However, with the ability to *backtrack* to parent activities through the context view, users were able to obtain the snapshot on any point along that activity trail. This makes it easy for them to review and check back on what they have done.

To put the concept of the the activity trail into a broader perspective, suppose you are editing a document. As part of this activity, you may have performed some internet searches to find out some information. These searches are not tasks or milestones that you set beforehand, but are activities that dynamically arose as you edit that document. They are therefore an integral part in the evolution of that activity — they form part of the activity trail of your current activity. Simply knowing that you have done those search activities can allow you to make better sense of where you are as far as editing that document is concerned. Other forms of activities that would form part of that activity trail may include past email exchanges regarding that document, and the versioning history of that document. Unfortunately, current computing paradigms simply do not capture this perspective in an activity-centric way. The UMEA system captures some of this characteristics, but in a project-specific way.

For an activity like arranging a meeting, there are several pieces of information that invariably help to provide context to the activity, like who originated the meeting, who is expected to attend, what has been arranged etc. Users felt that the ADC model is useful in *gathering information* for the activity-at-hand. The relevant tasks, events, and messages — all the things that one is doing in relation to the activity-at-hand — are grouped together as a context under the current activity automatically. By relieving the user the burden of searching for such information, one user noted how it can be a time-saver.

> *I don't have to do any search, or use any memory, try to remember what I've done, what I haven't done. That would be a waste of time.*

Anything that requires you to go back and search through all your folders is a waste of time.

As humans, we often forget things, especially when our activities are done over an extended period of time, or when we get inundated with a large number of activities. In forgetting what we have done, we reduce the certainty in doing the things that we need to do. In forgetting how activities are related to each other, we reduce the amount of information on-hand to perform the current activity. One user noted this exact problem, and commented how the ADC model is useful as a *memory aid* to help overcome such memory lapses.

I don't know all the SMSes I sent, tasks, projects, are related or not. So I have to remember that they are related, and look for them. If it is done in a 13 step thing, that is, done over a period of time, it will be forgotten. If it's not put together, I may have to put in additional tasks which is to tell Paul, tell Sally, instead of having it put automatically into things I've done. So I don't have to think back, have I told people or not?

Another user found the ADC model useful in reminding them that there is a group of related activities that needs to be looked into when they are at any particular activity. As such, it can help to at least avoid missing out on certain tasks.

Activities associated in a parent-child relationship with the activity-at-hand were presented in a list, where their relationships with the activity-at-hand were denoted by icons representing parent, child or sibling. In this way, users felt that there was more clarity in how the activity was started (i.e., the parent), and what actions have occurred afterwards or have yet to be completed (i.e., the siblings or children). One user felt that in term of *interpreting information*, this was a powerful way to figure out immediately what the activity is all about.

[At] a glance, it provides powerful information, what I have done, what was the activity about, and straightaway I know what is the milestones that needs to be accomplished. It's a powerful presentation of the activities summary.

As a *mental aid*, the ADC model can help the users in several ways. One user commented how it resembled a good-old trusty tool that he used.

It's exactly how I do my stuff, except that it's on a piece of paper.

Another user described how the ADC model was able to help him mentally map out things that he needed to do. He felt that this has always been a much needed capability, but one that has not been available before.

[In] your mind, you have a more systematic idea of what you need to do. Here, at least you can go back to a parent task, and from the parent task, you will be able to do the subtasks that you are required to do. It kind of tells you that mentally in your mind, you know what to do, but when you relate it to a machine, you want the machine to aid you in your thinking, how you want to do it. Each time you finish one task, you know when or which subtask or parent task that you need to do, to check back. In what I have used before, this capability is not present.

Finally, users found the ADC model's ability to tie activities together to be a useful tool for *organizing activities*, such as in helping them sub-task the things to do for a particular main task. One user commented that it is like having a virtual thread to tie activities together:

In a nutshell, what it does for you is like giving you a virtual thread between all the different tasks or events or whatever it is that you are trying to keep track of. It allows you to virtually ties things together, and gives you pointers to different elements of a task-at-hand.

Providing easy access to the different elements that make up the activity makes it easier to find things in relation to each other, except that the user does not have to do the heavy lifting of searching through mountains of data by himself.

4. Limiting Factors

On the flip side, users pointed out several limiting factors to the extent in which activities as context can be perceived as useful.

The first is the *user interface* used to convey the relationship between activities and to interact with activities. For example, in the current implementation, related activities are presented simply as a list, sorted by activity type. Some users felt that this is inadequate, and that there should be more hierarchy or structure to the presentation, or alternatively, to have the ability to expand or collapse selected portions of the list.

> *In the current form, I don't find it useful, so I rate it as a 2. It's a user interface issue. But if the interface is improved, the idea of it seems useful, say 5... I think it will be very good to have a very clear distinction on the hierarchy. Right now, everything is one line. If it's arranged in terms of when it's being done, in terms of a hierarchy, it will be easier to look at.*

Clearly, effective context presentation in a confined space, such as on a mobile device, is an area for much needed research.

Another limiting factor is the *degree of burden* perceived by users who manually labeled their activities. Some users commented that there is one too many steps in the current implementation. Others felt that it was a mental chore to have to decide on how to label the new activities, especially when there is a lack of familiarity in the concepts of parent, child, and sibling relationships. In the worst case, it may turn the ADC model into a hindrance rather than a helpful tool, as noted by one user:

> *There is a downside that using this additional function requires me to make an additional level of thinking to relate the activities beforehand. So it could be a double-edged sword. If I actually did conscientiously tried to relate the activities, put in the necessary thinking process, then this view function would be useful to me. If I had not cared, put everything down as unrelated, then this would be counter-productive later when I revisit the activities. It can be an additional layer of burden.*

The issue of automatic labeling against manual labeling will be revisited in a later part of the chapter.

The final limiting factor is the *degree of flexibility* that a user has in using the ADC model to manage and interact with his activities. If using the ADC model is a choice that the user can make, then it may be perceived as less of a hindrance when it does not perform to the user's expectations, or when he is still learning how to maximize its potential and utility. One user described how it was reassuring that, in the current implementation, he can always choose to go back to the conventional way of interacting with his messages, events, and tasks, whenever he finds difficulty with the new methodology.

> *It was backwards compatible, so that was good. If I get stuck, I can just go back to the old way. I'm presented with more options, so that increases the complexity, especially first time using it. With the more options, there were a couple of times that I got stuck, I wasn't sure how to get back to my root. I was able to click "Home" and go to the event, or whatever it was. I wasn't constrained to having to do it this way.*

B. PERCEIVED IMPROVEMENT

In terms of perceived improvement, the goal of the analysis is to find out if the users felt that ADC model improves their mobile computing experience. For example, does it make it easier or more efficient in interacting with multiple activities? Does it make information more easily accessible?

1. Quantitative Analysis

In general, users found that the ADC model improves their computing experience (cf Figure 19). On a scale of 1 to 7 — 1 being no improvement, 4 being some improvement, and 7 being significant improvement — users gave an above-average score of 5.13 (coincidentally the same score as for perceived usefulness).

The standard deviation of 1.74 indicate some variability in users' opinions. Nevertheless 87% of the users indicated that there was "some improvement" or better over current methods, with close to half the users (47%) giving it a high rating of 6 or 7 (cf. Figure 19). So just like the case in perceived usefulness, despite some apparent variability in the data, users' response was clearly in the positive.

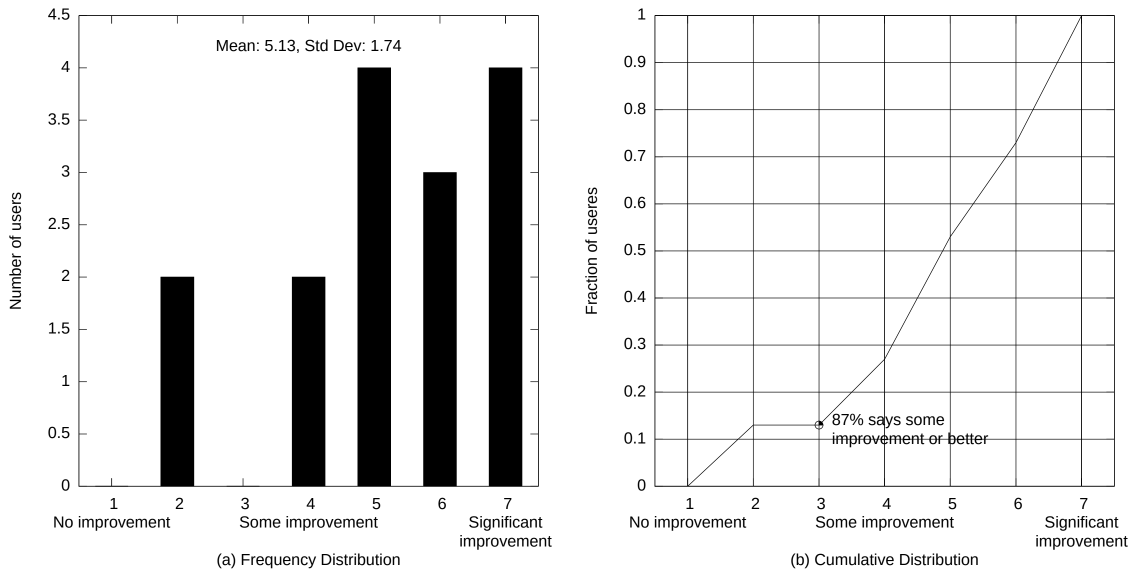


Figure 19. Perceived improvement over current methods.

2. Categories of Improvement

The method used in the analysis here is exactly the same as in the previous section. Again, it must be emphasized that the categories of improvement listed here are not categories chosen by the researcher beforehand, but were derived from open-ended answers provided by users in the interviews using the open coding technique.

The main improvement that the majority of the users identified with immediately is in the ability to get *information on-hand* for the current activity. Again, this goes back to realizing the benefits of an associative model of information access. Users were able to see the tree of related activities (the originating activity, related messages, the subtasks etc) at one screen, rather than having to go back to the home screen and juggle between applications to look them up again. Having a link back to the parent activity also lets the user better *keep track* of what he is supposed to do, and the odds and ends that needs to be cleaned up. There is also now a better *organization* and structure to one's activities. One user commented on how this has made it easier to close a communication loop:

The main difference is that for the task that I'm doing, all the various

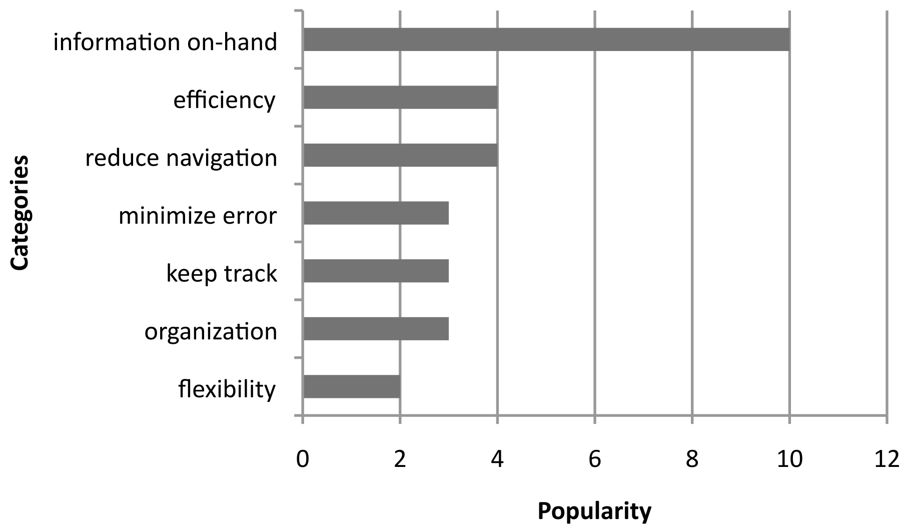


Figure 20. Categories of perceived improvements over current methods. Each category has a popularity score, which reflects the number of users whose remarks fall under that category.

> *actions that I have taken, are kept together within one glance, one look space. . . I am able to see all the related substasks and what I have done so far. Especially when I need to reply to John, telling him that the meeting has been set up, I can just go there and click and reply him, instead of having to go back to my SMS list, and search among ten thousand other SMS messages, and don't know which John message I have to look for. So just looking at this one, I can find which one.*

Having information on-hand can in turn then have several other related impacts on the users' computing experience. First, it improves the *efficiency* with which a sequence of activities is carried out. The number of steps between tasks is reduced, due to *reduced navigation* between screens. Interaction between activities becomes easier and quicker, since less time is spent in searching for related activities. One user recognized that this will make a difference when there are a lot of activities that a user needs to contend with:

> *It's just faster to carry out the next activity because it's linked. The difference will really come up when there's a lot more activities, more*

> *events... In terms of efficiency, it means it takes a shorter time to complete. Then you need less time to search for related messages, events. On a given normal phone, you may have 30 tasks and 20 events.*

Having information on-hand can help the user to avoid missing out on certain steps or inadvertently letting some things fall through the gaps. Consequently, one user noted how it can *minimize errors* and improve the certainty on actions that need be taken:

> *All the information are there at one glance. You've got all the information on-hand. Let's say when you reply, you won't say the wrong things. You may say it's set up, but you realize that out of 4 sub activities, one is not completed. and that screws up later. You've sent out the mail to say that it's done, but it's not actually done yet. You're more certain on the action to be taken.*

Finally, one user found increased *flexibility* in how he can interact with the phone. Conceptually, through the ADC model, he can choose to interact with the phone in terms of processes, instead of interacting with phone functions:

> *The process of trying to get my objective done... is much more smoother, as I don't feel restricted by the processes of the phone. Now I know that I've got other options in terms of navigating around this phone. I am able to feel at ease while I'm using the phone... This is definitely a more intuitive way to navigating, because ultimately, when a phone designer design a phone and sells the phone, the phone can perform a thousand and one wonderful things, but ultimately, it has to cater to the user's needs, the processes. It's what the user is looking for, flexibility in the process.*

3. Limiting Factors

On the flip side, the main complaint expressed by users was the *non-intuitive interface* in interacting with activities and the *additional steps* required in labeling the activities, for users who manually labeled the activities. For example, one user commented on how it has become more confusing to navigate the system because the interface was not clear on what he needs to do:

In terms of navigation, initially it was a bit confusing, because it's not intuitive to click on it for the task that you want to do. For example, you're supposed to have a subtask to that message. Because you do not know that is the right link for the task, so you can't tell those are the things you need to click. There should be icons to tell the user to click on... something like prompts for actions.

Another limiting factor is the experimental setup. One user, who gave a low improvement rating despite a high usefulness rating, felt that the trial scenario is too simple to be able to discern the improvements while facing a learning curve over the new method.

The tasking given is too simple. On one hand, you're trying to figure out the buttons. On the other hand, this tasking is not exactly that big for you to realize the significance, whether the process has improved. You're also trying to get used to the workflow.

This is basically a limitation of the experimental setup, and can be alleviated if the study is extended to an embedded testing environment where the users have more time to evaluate the model under more diverse settings.

C. LABELING OF ACTIVITIES

There are some contextual relationships between activities that require some subjective properties to be attached to activities. One example is the set of activities that are related by common tags. Another example is activities that are related by priorities.

For clarity, the term "label" will be used here as a general term in place of "subjective property", while the term "labeling" is used to denote the action of attaching the label to the activity.

The current implementation of the ADC model requires the labeling of newly created activities in relation to the current activity, that is whether to label it as a new child activity or a new sibling activity. This piece of information is then used to construct a parent-child relationship between activities as a context to the activity-at-hand, when it queries the system for context. The details of this process has already been described in an earlier chapter.

The labeling can be done either manually by the user, or automatically by the system on behalf of the user. This raises some interesting questions:

- Is it always better to automatically label activities? There is a natural tendency to assume that automatic labeling would yield better satisfaction for the users, since the users are relieved of the physical and mental burden of doing the work. Is there any justification for manual labeling of activities?
- If activities are to be automatically labeled, what are some rule-of-thumbs that can be applied? Are there any tendencies that users appear to favour, or a structure that is more likely to emerge?
- If activities are to be manually labeled, what are some reasonable principles that can be followed? How can one achieve some balance between manual labeling and automatic labeling?

This section attempts to address these questions by analyzing data collected from the evaluation. Out of a total of 15 users who participated in the evaluation, 10 users manually labeled their activities. The remaining 5 users let the system automatically label their activities.

1. Heuristics for Automatic Labeling

Table 1 shows the initial rule-of-thumbs that were used to automatically label activities in the evaluation. These are pair-wise relationships between messages, events, and tasks that were derived out of common sense more than anything. We want to see how well these "best-guesses" hold up against data from users who manually labeled their activities.

		From Activity		
To Activity		**Message**	**Task**	**Event**
Message	child	20	8	5
	sibling	7	0	0
Task	child	5	3	4
	sibling	3	0	0
Event	child	3	0	0
	sibling	0	4	0

Table 3. Aggregated child and sibling relationship counts from users who manually labelled activities.

Table 3 is derived by simply counting the respective pair-wise relationships of how users labelled their activities. Comparing it to Table 1, the following observations can be drawn:

(a) The initial "best-guesses" match up very well with user data. Table 1 turned out to be a good set of heuristics to use for automatic labeling of activities in this evaluation after all.

(b) Some pair-wise relationships between activities have specific semantics that appear to dictate how they are labeled. Specifically, message replies are always labeled as child activities, while forwarded messages are always labeled as sibling activities. A subtask is obviously always labeled as a child of a main task.

(c) There appears to be a general tendency for users to favour labeling a new activity as a child rather than as a sibling.

The key implication of the first observation is that the "truthfulness" of the heuristics should not be a factor that affects how a user who had activities automatically labeled perceives usefulness and improvement. In other words, it is unlikely that, for instance, if the user gave a poor rating, it is because the auto-labeling heuristics failed him. The upshot is that data from users who had activities automatically labeled can be compared on equal footing with users who manually labeled their activities. This is an important result going into the next section when we analyze the differences in perceived usefulness and perceived improvement for automatic versus manual labeling.

The second observation indicates that specific pair-wise semantics must be considered when populating the heuristics table. Exceptions must be made where applicable.

The last observation warrants some further examinations. What is shown in Table 3 are aggregate counts. As with all aggregates and averages, the caveat is that the situation behind individual pair-wise relationships is taken out of consideration. It has been pointed out that some pair-wise relationships between activities have specific semantics that dictate how new activities are labeled. So the last observation could just be a result of more situations that dictate the labeling of new activities as child activities than as siblings.

To examine the claim in more details, two situations in the evaluation script (cf. appendix A) were identified that required the user to make conscious decisions on how to label a new activity, that is, whether it is to be labeled as a child or a sibling. Either way is possible, and is entirely up to the user at that point.

These two situations then become mini case studies. Each case involves three activities:

- Case 1 revolves around steps 2, 3 and 4 in the evaluation script. The activities involved are a message, a task and an event.
- Case 2 revolves around steps 5a and 5c. The activities involved are a task, a message, and then another task.

Two indicators are then designed to evaluate the cases:

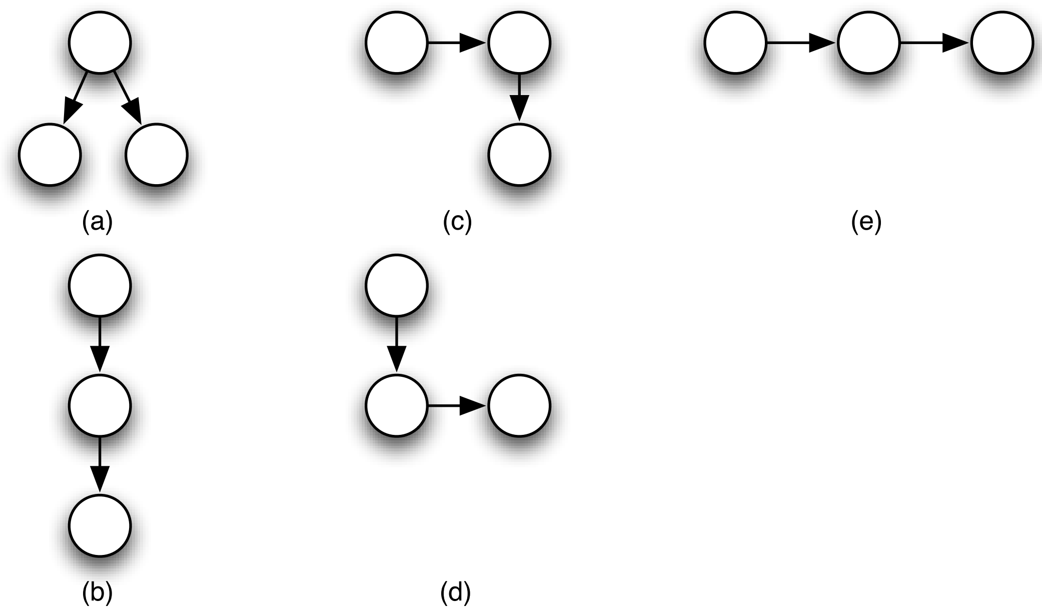


Figure 21. Possible relationship patterns between three activities. Patterns (a) and (b) are termed as child patterns, while patterns (c), (d), and (e) are termed as non-child patterns. Downward arrows indicate creation of child nodes, while lateral arrows indicate creation of sibling nodes.

	Indicator 1		Indicator 2	
	Child activity	Sibling activity	Child pattern	Non-child pattern
Case 1	8	5	3	3
Case 2	17	2	7	2
Total	25	7	10	5
Ratio	0.78	0.22	0.67	0.33

Table 4. Data showing the likelihood of users to label activities as child activities or sibling activities.

- Indicator 1 is a first-order indicator that simply counts the number of child nodes versus the number of sibling nodes in each of the two cases. The hypothesis here is that if the number of child nodes exceeds the number of sibling nodes, then that indicates a tendency for creating a new activity as a child.

- Indicator 2 is a second-order indicator that examines the structural patterns of the three activities involved in each of the two cases. It was observed that there are five possible patterns that these three activities can make (cf. Figure 21). Patterns (a) and (b) are termed as child patterns, because they are dominated by child nodes. Patterns (c), (d), and (e) are termed as non-child patterns for the opposite reason. For each of the two cases, the number of child patterns versus non-child patterns are counted. The hypothesis here is that if there are more child patterns than non-child patterns, then that indicates a tendency for creating a new activity as a child.

The result of the analysis is shown is Table 4. For indicator 1, there is a 78% likelihood that a user would label an activity as a child. For indicator 2, there is a 67% likelihood that a user would label an activity as a child. Thus, there is evidence from the data to support the claim that there is a tendency for users to favour labeling a new activity as a child rather than as a sibling.

This result can be used to set labels for pair-wise activity relationships that is not yet validated by data or where semantics are unclear. Thus, coupled with the need to make exceptions for pair-wise semantics, we can derive a two-step heuristic for labeling activities as follows:

Step 1. In general, label the new activity as a child.
Step 2. If a specific semantic exists for the activity to be labeled either as a sibling or as no relation, make an exception for it.

For example, in the initial heuristics table (cf. Table 1), an event-event relationship was to be labeled as a sibling as a matter of "best-guess". But in the

	From Activity		
To Activity	**Message**	**Task**	**Event**
Message	child Forward: sibling	child	child
Task	child	child	child
Event	child	sibling	child* Recurring: sibling*

Table 5. Modified heuristics for auto-labeling of new activities. *Note the modifications made to the Event-Event relationship.

evaluation scenario, there is no event-event relationships that are created by users to validate the guess. However, using the labeling algorithm, we can update the event-event relationship in the heuristics table, as shown in Table 5. The event-event relationship has been modified from a sibling to a child, except when the new event is a recurring event. Now, the truthfulness of this mapping will need to be validated separately.

The two-step labeling heuristic can also be used when a new activity class is to be added to the table, and the respective pair-wise relationships need to be populated on a first-cut basis.

In general, the logic behind the two-step heuristic should be adaptable for other labels as well, such as priorities.

2. Automatic Versus Manual Labeling

There exists an old adage: "automatic for comfort, manual for performance." This, of course, refers to the debate over which transmission system is better for driving a car — automatic transmission or manual transmission? Proponents of automatic transmission say that it is easier to operate, uses only one of your feet, and allows you to keep both hands on the steering wheel. Proponents of manual transmission claim better performance in terms of power and mileage, and more control in tougher driving conditions.

	Usefulness		**Improvement**	
	Mean	Std. Dev.	Mean	Std. Dev.
Manual labeling	4.85	1.80	5.10	2.04
Auto labeling	5.70	1.10	5.20	1.10

Table 6. Means and standard deviations for perceived usefulness and improvement by users who manually labeled their activities, and users who allow the system to automatically label their activities.

The same "automatic or manual" debate can be made over the how an activity is labeled, regardless of whether as a child or a sibling in this case. It is easy to assume that most people would prefer automatic labeling, as it relieves them of the burden of doing the work, and so automatic labeling should be the way to go. Indeed, we have even made the observation previously that the degree of burden in manually labeling activities can be a limiting factor towards perceived usefulness.

But is this assumption borne out by user data? What are the relative merits of both approaches? Is there ground to pursue a balanced approach?

To investigate this issue, the data of the users who manually labeled their activities are segregated from those who let the system automatically label their activities. The data is then analyzed along the dimensions of perceived usefulness and perceived improvement.

Table 6 gives the statistical mean and standard deviation indicators, while Figure 22 give a visual chart of the data distribution.

From the table and the chart, the following observations can be made:

(a) All the users who had their activities automatically labeled perceived the ADC model to be at least somewhat useful, and offered at least some improvement. All the negative sentiments in perceived usefulness and perceived improvement came from users who did manual labeling.

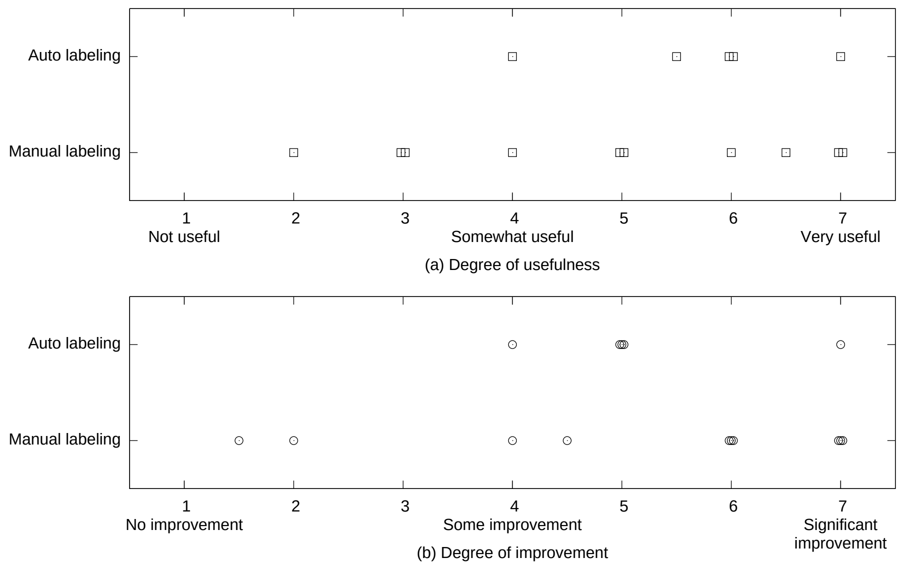


Figure 22. Perceived usefulness and improvement by users who manually labeled their activities, and users who allow the system to automatically label their activities.

(b) As a result, the mean scores for perceived usefulness and perceived improvement is generally higher for automatic labeling than for manual labeling. For perceived usefulness, it is significantly higher, from 4.85 to 5.70. However, for perceived improvement, it is only marginally higher, from 5.10 to 5.20. For the latter case, one can even argue that the difference is statistically negligible.

(c) The standard deviations of 1.8 for users who did manual labeling indicates the variability in their opinions of activities as context. The tighter standard deviation of 1.1 for users whose activities are automatically labeled is a simple reflection of the first observation.

(d) 60% of the users who did manual labeling rated perceived improvement to be significant, i.e., a 6 or 7. These scores were able to offset the low scores given by other users. On the other hand, only 20% of the users whose activities were automatically labeled rated perceived improvement as significant. In other words, a higher percentage of users who manually labeled their activities found huge improvement than those whose activities were automatically labeled.

The first observation is easy to explain. For users whose activities were automatically labeled, they got additional benefits for no additional effort. In this case, it was something that they found useful and offered improvements over what they have. Hence the absence of negative sentiments from this group of users. For users who manually labeled their activities, they have to balance the additional physical and mental effort against perceived usefulness and improvement. Some of them clearly do not see the benefits as worth the while, or the benefits are not apparent enough. Hence some negative ratings coming from them.

The last observation is more interesting, and can be explained by considering two factors: gratification, and control. When a user puts in an effort to do something, immediately an expectation is formed, and he gets gratification when something good comes out of it. Consequently, he is in a frame of mind to be able to appreciate the returns more and understand its potential impact better. One user described how he can feel a sense of reward after the additional effort he put in, which immediately let him observe unique benefits of new tool:

> *In terms of the additional thinking process required, in terms of relating those activities, if you balance that out as whole, on one hand I have to think an additional step, but of course I get rewarded by the fact that I get all the activities related to me. I get to see it at a glance. This helps me avoid missing out on certain steps and some other things that will fall through the gaps.*

Another user described how by manual labeling, he can create a way of working with the system that is intuitive to him, and gives him a higher degree of control in terms of what works best for him. Again, the upshot is a better appreciation of the returns and its potential impact on his working processes.

> *Although I was fumbling around with, do I make this a task or subtask or co-task or what have you. By asking the user to create it explicitly, it allows the user to come up with his own system, and that lends itself to a system that is intuitive to me...I would probably fumble with it the first few times, but I would figure out what works best for me and*

then that would probably be the way I do it forever more. Others using the same system, at the "water-cooler" if you will, discuss this new capability, and see how everyone else is using it, to maybe shortcut that process.

The conclusion that can be drawn here is that while users can always appreciate the system doing the work of labeling activities for them, it will make the most difference to them if it is combined with some level of user involvement. Thus, there is sufficient grounds for further research into a balanced approach to labeling. By a balanced approach, we mean a combination of manual and automatic techniques.

D. A NOTE ON EMULATOR TESTING

The study was conducted on a Google Android mobile phone emulator. For the most part, the emulator was adequate in simulating interactions on an actual device. However, during the course of the user trials, it was found that in terms of interacting with an emulator versus interacting with an actual device, there is one key difference that needs to be taken into consideration during emulator testing, because it may have an impact on the perceptions of certain aspects such as usability.

The difference is that on an actual device, the user uses his fingers to access various buttons and user interface elements on the device, whereas on the emulator, the user uses a pointing device, typically a mouse. From the user's point of view, there is some difference in expectations between pointing with a finger and pointing with a mouse, and this may give rise to some problems when testing on an emulator.

It was noted that there is a tendency for some users to double-click on an icon on the emulator. Double-clicking on an icon to activate it is the normal expectation on a desktop computer. But on a touchscreen mobile device, a single tap is typically the norm. The problem with double-clicking with the mouse is that it may be registered as two different taps on the emulator. So if the icon is a delete icon, then that may cause two delete operations to be inadvertently executed on two consecutive items on a list, causing disruptions to the trial process. If the icon activates screen transitions,

Categories of Usefulness	Categories of Improvement
Situation awareness	Information on-hand
Gathering information	Efficiency
Memory aid	Reduce Navigation
Interpreting information	Minimize Errors
Backtracking	Keep track
Mental aid	Organization
Organizing Activities	Flexibility

Table 7. A summary of categories of usefulness and improvement.

such as a forward or back button, then that may trigger two transitions instead of one, causing disorientation on the part of the user.

It is certainly possible to guard against "double-tap" within the software. A more elegant solution would be to use a touchscreen monitor with the emulator instead, rather than using a mouse to interact with the emulator. This way, the tendency to double-click can be greatly reduced or even eliminated, so that there is little chance to the occurrence of the "double-tap".

E. SUMMARY

This chapter presented the methods and results arising from the analysis of the data collected from the user evaluation of the ADC model. In general, users found that the ADC model not only provides improvement over what they currently have, but also provides unique benefits for their computing experience. The categories of perceived usefulness and perceived improvement are summarized in Table 7. These categories can be used as measures to populate survey forms for future larger-scale studies.

Furthermore, analysis of the labeling practices of the users reveal that users tend to label new activities as child rather than siblings. This result gives rise to a two-step heuristic algorithm that can be used on a first-cut basis to automatically label newly created activities.

Results also show that while automatic labeling is invariably popular with users, users who manually label their activities expressed a greater degree of perceived improvements over users who had activities automatically labeled for them. Hence, there is sufficient grounds for further research into a balanced approach to labeling.

THIS PAGE INTENTIONALLY LEFT BLANK

VII. CONCLUSIONS

In painting his vision for ubiquitous computing, Mark Weiser described how technologies should become "invisible" in order to be ultimately useful [67]. That is not to mean that technological artifacts should disappear from sight, but rather that technology be so integrated into the way we do things that they become essentially unnoticed. From that perspective, context-awareness is in fact not the desired goal; context-unawareness is. This can only happen when context-awareness has become an integral part of our everyday mobile computing activities. Context thus becomes as much as an outcome as it is a premise. We begin to task in context.

The emergence of the mobile device, with all the advanced technical capabilities and social functions that it offers, provides the perfect conduit to realize this vision. However, for this to happen, there is a need to re-orientate our research efforts in mobile computing towards a notion of context that is more in line with the human experience, as well as to incorporate context-awareness into the very essence of mobile computing.

Prior research in context-awareness has largely been dominated by a positivist notion of context, where analysis tends to be objective, qualitative, and predictive. While this notion of context is sufficient for well-defined and focused applications, it suffers from two main shortcomings. First, it fails to consider context as a dynamic construct that arises from a user's interactions. Second, it lacks enough consideration for the role of the human actor in context-awareness. As a result, it is inadequate for dealing with the kind of high-level activities that people naturally engage in as part of their everyday lives, such as "write a paper", "arrange a meeting with team", "attend Bob's birthday" etc. For such scenarios, the contextual set is much more dynamic and unpredictable, and highly dependent on the person's activity at that time.

In this dissertation, an activity-driven model for an interactional notion of context has been proposed to addresses these shortcomings. To validate the model, a prototype implementation of this model has been developed on the Google Android mobile phone platform. A user evaluation of the model was conducted using the prototype running on an emulator.

Reviewing the research questions that were posed in the introduction of this dissertation regarding the model:

1. *Does the activity-driven model of context provides utility and improvements to the mobile user for his everyday activities?*
   When applied to the personal information management (PIM) domain utilizing a parent-child relationship between activities as the principle context, it was found that the activity-driven model of context does indeed provide utility to user's computing experience as well as improvement over what he currently has.
2. *In what way is the model useful and better than what is currently available?*
   Using qualitative analysis techniques, it was found that the activity-driven model of context was able to offer users unique benefits such as situation awareness, and memory and mental aid, as well as improvements due to an associative model of information access that resulted in greater efficiency and minimized errors. These are capabilities that have been lacking for the mobile users before this model.

From these results, we can conclude that the activity-driven model of context has been validated through user evaluation.

The results also validated the novel methods described in this dissertation. The mediator approach to context gathering among activities was validated by a prototype implementation and evaluation on the Google Android mobile phone platform. The rule-based discovery of parent-child relationships between activities was validated via the actual labeling of activities by users.

Additional results were also obtained. Analysis of the labeling practices of the users gave rise to heuristics that can be used on a first-cut basis to automatically label newly created activities. Results also show that while automatic labeling is invariably popular with users, users who manually label their activities expressed a greater degree of perceived improvements over users who had activities automatically labeled for them. This indicates a need for further research into a balanced approach to labeling activities that incorporate both automatic and manual techniques.

A. DISCUSSIONS

1. Associative Model of Information Access

One advantage to our model of context is that it enables an associative model for information access. In the dinner plans example discussed previously, information such as the parties involved and their contact information, the location of the restaurant and its contact information etc, are made available to the activity-at-hand because they are naturally part of the various activities that form the context to that activity. In this way, the user has them immediately available, instead of having to dig through hierarchies of lists and folders for them.

Abowd and Mynatt [55] pointed out that while hierarchical models of information are a good match for well-defined tasks, models of information for activities are principally associative, since information is often reused on multiple occasions, from multiple perspectives. They argued that associative and context-rich models of organization support activities by allowing the user to reacquire the information from numerous points of view. These views are inherent in the need to resume an activity in many ways, such as to remember information relative to other current information, e.g., a document last edited some weeks ago or the document that a colleague circulated about some similar topic.

So the basic idea is to make it possible to have all the material needed for an activity ready at hand, available with little or no mental overhead [4]. Furthermore, Garlan et al. [68] pointed out that the most precious resource in the computer system is no longer its processor, memory, disk, or network, but rather human attention. So in helping the the user gather and manage contextually relevant information spread out over multiple activities in an associative manner, he spends less time navigating the device looking for piecemeal information, and more time and attention on the activity itself. This can be a significant gain, especially on a mobile device, due to its limitations arising from user interface and power constraints.

2. Activity Awareness

Our model of context allows system designers to "turn the table" in thinking about the relationship betweeen activity, context, and physical situations. Take location-awareness for instance. Rather than thinking about how activities can benefit from being location-aware, we can instead think about how locations can be made meaningful by being activity-aware.

For example, one of the things that we can do with our mobile device nowadays is to bring up a map of our current location, perhaps annotated with available services in the area. Now, if we can put these information in the context of our various activities that are relevant to that location, such as previous communications, pending tasks, and events, then the location would make more sense to us, because we can rationalize it against our own activities. An illustration of this concept is shown in Figure 23. Knowing that we are near a grocery store would be more meaningful if we have a need to "do grocery shopping".

So our activities form a context surrounding that location that adds meaning to how we perceive that location. Such a context is dynamic in nature and differ from instance to instance, because our activities change as time progresses. The next time we revisit that location, our activities would have changed, and thus so does the context, and consequently how we make sense of that location.

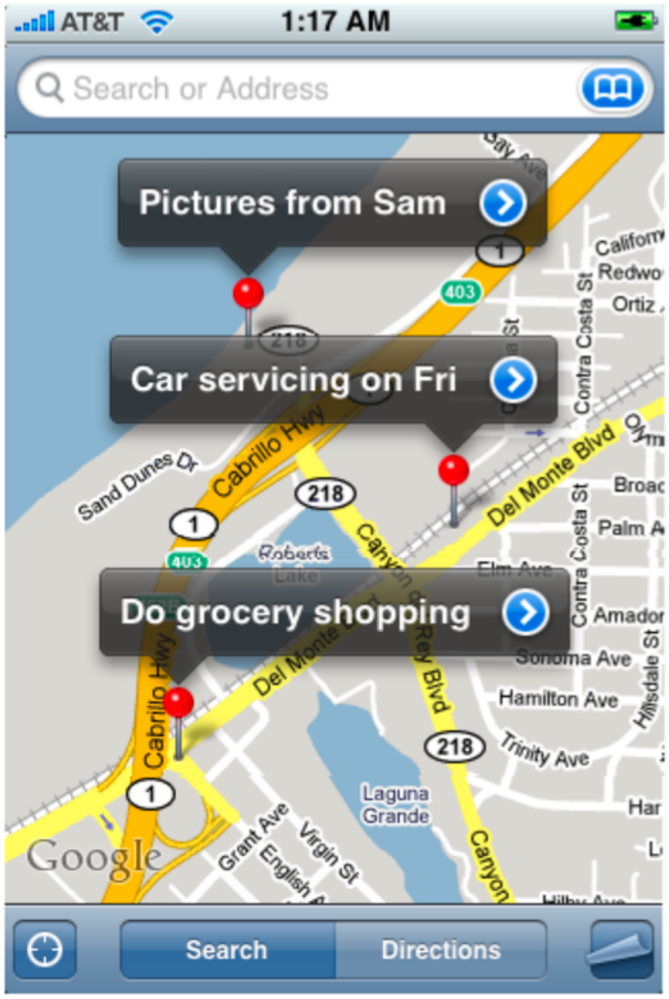


Figure 23. An illustration of an activity-aware map.

3. Context as a Dynamic Construct

One implication of the notion of activities as context is that the contextual set cannot now be determined a-priori. That means that the richness of contextual information is not tied down at the point of design, but can constantly evolve as the system and activities evolve. As a result, we begin to recognize and take advantage of context as a dynamic construct that arises from interactions between the user, the system, and the environment.

On the flip side, not being able to specify a contextual set a-priori can be a source of concern for the system designer, because it makes it difficult for him to determine the appropriate set of meaningful actions to perform on the user's behalf, which is often the goal of many context-aware systems. One solution to this conundrum is offered by Erickson [28]. He observed that even under the traditional positivist approach of context-aware computing, the ability of the system to recognize contexts and then act appropriately in a robust way is questionable. So rather than trying to take the human out of the control loop, he suggested to keep the human in the loop. Let the computer system do what it is good at: gathering and aggregating data, and let the human do what he is good at: making sense of context and determining what is the most appropriate to do.

In effect, this will promote a form of context-aware computing where the human is an integral part of the process, and where he is made to share some responsibilities in the decisions and actions taken.

4. Valued Information at the Right Time

Denning [69] described how technology is generating more information and at faster pace than our individual capacity to process it. Thus, much information is lost or ignored, resulting in users feeling overwhelmed, frustrated or detached. One way to deal with the problem is through a new approach called "valued information at the right time", or VIRT. The key to VIRT is deciding which information is of value and to whom. Once this is known, the distribution network can be then be set-up, for example in a push or pull configuration, to optimally deliver the information to the user.

But the question that remains unanswered is: how is the relevancy and value of information determined in the first place? Much of it depends on the context and intentions of the user. Thus, given the dynamic nature of context, research into the interplay between activity and context can give rise to a better sense of how relevancy of information can be determine dynamically, which will be crucial to the realization of VIRT.

B. LIMITATIONS

The limitations inherent to this study are as follows:

1. While metrics such as perceived usefulness and perceived ease-of-use are well understood and widely used in usability studies, the effects of perceived improvement has yet to be established, especially in relation to the other metrics. For example, one of the concerns that can be raised is this: how is perceived improvement correlated to perceived usefulness? Is it possible for a tool to be rated useful although no apparent improvement is perceived? Conversely, is it possible for a tool to be rated as an improvement if no obvious utility is perceived?
2. The evaluation was limited to a single scenario within a single domain. In this case, the domain is the personal information management domain, and the scenario is that of arranging a meeting.
3. The evaluation being conducted in a laboratory setting limited the amount of interleaving of activities and external interruptions that a mobile computing user would otherwise face in the real world.
4. The study focused principally on the parent-child relationship as the contextual relationship between activities.

C. RECOMMENDATIONS FOR FUTURE WORK

1. The obvious progression from this study, and probably the more immediate one, is to seek expand the model through controlled empirical trials. One of the main aims of subsequent studies should be to allow for such scenarios and context relationships to naturally arise from users' interactions, instead of being set at the onset of the experiment. In addition, objective measures such as time spent on the device can be collected to expand the model.

2. The scalability of the model needs further studies. As the mobile device becomes increasingly populated with activities due to use, two issues needs to be examined. First, with regards to context gathering using the mediator approach, what is the impact on the response time and processor load? Second, with regards to context interaction, as the list of contextually-relevant activities grows, how should the list be sorted and filtered to prevent the user from being overwhelmed with information?
3. A much-needed area of research is to investigate how the limited user-interface modes of a mobile device can be used effectively to not only inform the user of the context in relation to the activity, but also to facilitate interactions between them. The current implementation presented contextual activities simply as a list, but users have expressed that it is neither adequate nor optimal. Since context is a relation between activities, one research direction is to investigate how to display activities and context in a graphical manner similar to spatial hypertext [70]. This would allow the exploitation of the unique human ability to make sense of information in a compact spatial dimension.
4. The model can be extended in several ways. One possibility is the social networking variation (cf. Figure 24a). That is, how can the model be extended beyond the device to include context gathering across devices, thus by implication in a social circle? Another possibility is the cloud computing variation (cf. Figure 24b). More and more information is being stored in the cloud, and more computational resources are increasingly accessible from the cloud. So how can the model, with the benefit of an associative model of information access, be expanded to include this trend in cloud storage and cloud computing to further enhance the power of mobile devices?
5. The interactional notion of context should not be investigated in isolation of the positivist approach to context. If anything, they should be viewed as complementary instead of opposing views of context. There should be research

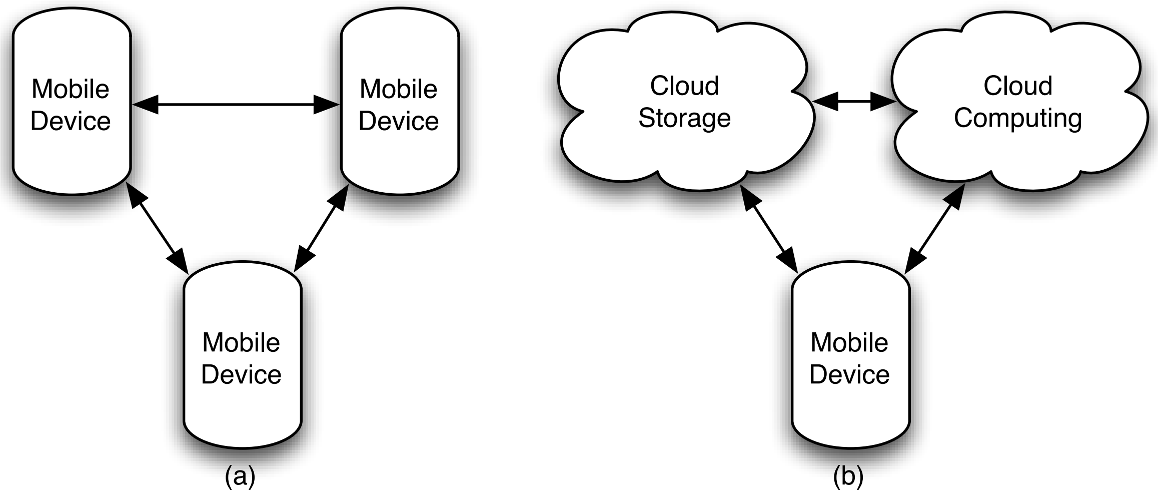


Figure 24. Two networked variations of the activity-driven model of context. Variation (a) is the social networking variation. Variation (b) is the cloud networking variation. Both variations should be targets for future research.

to see how these notions of context can exist in symbiosis. For example, how can a sensor-driven location-awareness be enriched with an activity-driven context? Or how can the contextual activity set be sorted or filtered with help from sensors or other cues?

6. Results from this study points to a balanced approach to activity labeling that combines automatic and manual techniques, for such subjective properties as child/sibling, tags, priorities etc. Further research is needed to propose and investigate workable models for this approach.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A. EVALUATION SCRIPT

1. Wait to receive a message from John. He will be asking to meet up with the project team one week later.
2. Reply to John, saying that you will arrange the meeting, and send confirmation to him later.
3. Create a task called Arrange meeting.
4. Create an event for the meeting, say one week later.
5. There are 3 sub-activities of the Arrange meeting task that you need to do.
   (a) The first sub-activity is to message Paul about the meeting.
   (b) The second sub-activity is to message Sally about the meeting.
   (c) The third sub-activity is to create a task is to book a room for the meeting.
6. Wait to receive a reply from Sally.
7. Assume that you have booked the room. Mark the task as done.
8. Locate the meeting event, and update the location of the meeting.
9. Wait to receive a reply from Paul.
10. When you think that the task Arrange meeting is complete, mark it as done.
11. Locate the original message from John, and reply to him that the meeting has been set up.
12. Return to the home screen.
13. One week has passed.... Locate the meeting event.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B. INTERVIEW PROTOCOL

INTRODUCTION

The purpose of this exercise is to evaluate the mobile computing techniques that we are investigating. The emphasis is not on the user interface, but on your perceptions on aspects such as interpreting information, ease of use, and efficiency.

This exercise consists of two parts. The two parts are mostly the same, except that in the second part, you will have access to some additional tools.

In each part, you will follow a script to perform a sequence of computing activities on a Google Android mobile phone emulator. These activities are in the Personal Information Management (PIM) domain. At certain points in the script, I will come in briefly to inject external inputs.

After each part, you will be interviewed. There is no right or wrong answer. Criticisms of the system are welcomed. The important thing is to be honest and expansive in your response.

You are also encouraged to think aloud. If you have any questions regarding the script or the interface, please feel free to ask me. It does not disrupt the process at all.

PRELIMINARY

1. What is your age?
2. What is your curriculum?
3. On a scale of 1-7, 1 being "not familiar", 4 being "somewhat familiar", and 7 being "very familiar", how would you rate your familiarity with mobile technology like smartphones and PDAs?

PART 1

4. On a scale of 1-7, 1 being “dissimilar”, 4 being “neutral”, and 7 being “similar”, how do you think the applications you just used are similar or dissimilar to the mobile PIM applications that you have used?

5. Tell me, as much as you can, what you know about the meeting. You may refer to your smartphone.

6. In your own words, describe what you have just seen and done.

7. What aspects of the experience do you think can be improved? For example, in terms of sense-making of the activities, interaction between activities, and ease of use?

PART 2

8. Tell me, as much as you can, what you know about the meeting. You may refer to your smartphone.

9. In your own words, describe what you have just seen and done.

10. Compared to the first part of the exercise, do you feel that the related activities presented to you provide useful information for your current activity? Rate on a scale of 1-7, 1 being “not useful”, 4 being “somewhat useful”, 7 being “very useful”. In what way was it useful or not useful?

11. What do you understand by the terms “context” and “context-awareness”?

12. Do you feel related activities presented to you in this part of the exercise fit your idea of context?

13. On a scale of 1-7, rate how much you agree or disagree with the following statement, 1 being “completely disagree”, 4 being “neutral”, 7 being “com-

pletely agree": Related activities presented to me provides useful context to my current activity.

14. What other contextual relationships between activities do you feel will be useful?

15. Compared to the first part of the exercise, do you feel that your experience with interacting with the activities has improved? Rate on a scale of 1-7, 1 being "no improvement", 4 being "some improvement", 7 being "significant improvement". In what way has it improved or not improved?

16. On a scale of 1-7, rate how much you agree or disagree with the following statement, 1 being "completely disagree", 4 being "neutral", 7 being "completely agree": Direct interaction with activities improves my management of multiple activities.

17. How else do you think your interaction with activities can be improved?

18. Finally, overall, what other suggestions do you have that you think can improve this evaluation exercise, or mobile computing experience in general?

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] L. Palen, M. Salzman, and E. Youngs, "Going wireless: behavior & practice of new mobile phone users," *Proceedings of the 2000 ACM conference on Computer supported cooperative work*, pp. 201–210, 2000.

[2] C. Licoppe and J. P. Heurtin, "Managing one's availability to telephone communication through mobile phones: A french case study of the development dynamics of mobile phone use," *Personal and Ubiquitous Computing*, vol. 5, pp. 99–108, 2001.

[3] R. Ling, ""We will be reached": the use of mobile telephony among norwegian youth," *Information Technology and People*, vol. 13, pp. 102–120, 2000.

[4] D. A. Norman, *The Invisible Computer: Why Good Products Can Fail, the Personal Computer Is So Complex, and Information Appliances Are the Solution.* The MIT Press, Aug. 1999.

[5] P. Dourish, "What we talk about when we talk about context," *Personal and Ubiquitous Computing*, vol. 8, pp. 19–30, 2004.

[6] A. K. Dey, G. D. Abowd, and D. Salber, "A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications," *Human-Computer Interaction*, vol. 16, pp. 97–166, 2001.

[7] R. Want, "You are your cell phone," *IEEE Pervasive Computing*, vol. 7, no. 2, pp. 2–4, 2008.

[8] A. Cypher, "The structure of users' activities," in *User Centered System Design.* Lawrence Erlbaum Associates, 1986, pp. 243–263.

[9] H. S. Teo, "An activity-driven model for context-awareness in mobile computing," in *Proceedings of the 10th international conference on Human computer interaction with mobile devices and services.* Amsterdam, The Netherlands: ACM, Sep. 2008, pp. 545–546.

[10] H. S. Teo, "Activities as context: A mediator architecture for an interactional notion of context," in *Poster session at the Tenth IEEE Workshop on Mobile Computing Systems & Applications.* Santa Cruz, CA: IEEE, Feb. 2009.

[11] B. Schilit, N. Adams, and R. Want, "Context-aware computing applications," in *Mobile Computing Systems and Applications, 1994. Proceedings., Workshop on*, 1994, pp. 85–90.

[12] T. P. Moran and P. Dourish, "Introduction to this special issue on context-aware computing," *Human-Computer Interaction*, vol. 16, pp. 87–95, 2001.

[13] G. Chen and D. Kotz, "A survey of context-aware mobile computing research," *Dartmouth Computer Science Technical Report*, 2000.

[14] P. Lucas, "Mobile devices and mobile data-issues of identity and reference," *Human-Computer Interaction*, vol. 16, pp. 323–336, 2001.

[15] T. Winograd, "Architectures for context," *Human-Computer Interaction*, vol. 16, no. 2, 3 & 4, pp. 401–419, 2001.

[16] T. Hofer, W. Schwinger, M. Pichler, G. Leonhartsberger, J. Altmann, and W. Retschitzegger, "Context-awareness on mobile devices - the Hydrogen approach," in *Proceedings of the 36th Annual Hawaii International Conference on System Sciences*, 2002, p. 292302.

[17] J. Hightower, B. Brumitt, and G. Borriello, "The location stack: a layered model for location in ubiquitous computing," in *Mobile Computing Systems and Applications, 2002. Proceedings Fourth IEEE Workshop on*, 2002, pp. 22–28.

[18] M. Baldauf, S. Dustdar, and F. Rosenberg, "A survey on context-aware systems," *International Journal of Ad Hoc and Ubiquitous Computing*, vol. 2, no. 4, pp. 263–277, 2007.

[19] D. Svanæs, "Context-aware technology: A phenomenological perspective," *Human-Computer Interaction*, vol. 16, pp. 379–400, 2001.

[20] S. Greenberg, "Context as a dynamic construct," *Human-Computer Interaction*, vol. 16, pp. 257–268, 2001.

[21] L. A. Suchman, *Plans and situated action.* Cambridge Univ. Pr, 1990.

[22] B. A. Nardi, *Context and Consciousness: Activity Theory and Human-Computer Interaction.* The MIT Press, Nov. 1995.

[23] V. Kaptelinin and B. A. Nardi, *Acting with Technology: Activity Theory and Interaction Design.* The MIT Press, Oct. 2006.

[24] G. Fitzpatrick, W. J. Tolone, and S. M. Kaplan, "Work, locales and distributed social worlds," *Proceedings of the fourth conference on European Conference on Computer-Supported Cooperative Work*, pp. 1–16, 1995.

[25] G. Fitzpatrick, T. Mansfield, and S. M. Kaplan, "Locales framework: exploring foundations for collaboration support," *Computer-Human Interaction, 1996. Proceedings., Sixth Australian Conference on*, pp. 34–41, 1996.

[26] G. Fitzpatrick, S. Kaplan, and T. Mansfield, “Applying the locales framework to understanding and designing,” in *Computer Human Interaction Conference, 1998. Proceedings. 1998 Australasian*, 1998, pp. 122–129.

[27] V. Bellotti and K. Edwards, “Intelligibility and accountability: Human considerations in Context-Aware systems,” *Human-Computer Interaction*, vol. 16, no. 2, 3 & 4, pp. 193–212, 2001.

[28] T. Erickson, “Some problems with the notion of context-aware computing,” *Communications of the ACM*, vol. 45, no. 2, pp. 102–104, 2002.

[29] M. Chalmers, “A historical view of context,” *Computer Supported Cooperative Work (CSCW)*, vol. 13, pp. 223–247, 2004.

[30] V. Kaptelinin, “UMEA: translating interaction histories into project contexts,” in *Proceedings of the SIGCHI conference on Human factors in computing systems.* Ft. Lauderdale, Florida, USA: ACM, 2003, pp. 353–360.

[31] A. Krishnan and S. Jones, “TimeSpace: activity-based temporal visualisation of personal information spaces,” *Personal Ubiquitous Comput.*, vol. 9, no. 1, pp. 46–65, 2005.

[32] R. Beale and P. Lonsdale, “Mobile context aware systems: the intelligence to support tasks and effectively utilise resources,” *Lecture notes in computer science*, pp. 240–251, 2004.

[33] B. Brown, M. Chalmers, M. Bell, M. Hall, I. MacColl, and P. Rudman, “Sharing the square: collaborative leisure in the city streets,” in *Proceedings of the ninth conference on European Conference on Computer Supported Cooperative Work.* Springer-Verlag New York, Inc. New York, NY, USA, 2005, pp. 427–447.

[34] D. A. H. Jr and S. Card, “Rooms: the use of multiple virtual workspaces to reduce space contention in a window-based graphical user interface,” *ACM Transactions on Graphics (TOG)*, vol. 5, pp. 211–243, 1986.

[35] B. MacIntyre, E. D. Mynatt, S. Voida, K. M. Hansen, J. Tullio, and G. M. Corso, “Support for multitasking and background awareness using interactive peripheral displays,” in *Proceedings of the 14th annual ACM symposium on User interface software and technology.* Orlando, Florida: ACM, 2001, pp. 41–50.

[36] J. Bardram, J. Bunde-Pedersen, and M. Soegaard, “Support for activity-based computing in a personal computing operating system,” *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pp. 211–220, 2006.

[37] J. E. Bardram, “Activity-based computing–Lessons learned and open issues,” *Position paper a ECSCW 2005 workshop, Activity–From a theoretical to a computational construct. Paris, September*, 2005.

[38] E. Bardram, "Activity-based computing: support for mobility and collaboration in ubiquitous computing," *Personal and Ubiquitous Computing*, vol. 9, no. 5, pp. 312–322, 2005.

[39] A. K. Dey, "Providing architectural support for building context-aware applications," Ph.D. thesis, Georgia Institute of Technology, 2000.

[40] J. I. Hong and J. A. Landay, "An infrastructure approach to context-aware computing," *Human-Computer Interaction*, vol. 16, no. 2, pp. 287–303, 2001.

[41] D. A. Norman, *Cognitive engineering.* Lawrence Erlbaum Associates, 1986, pp. 31–61.

[42] B. Budge, "Pinball construction set [Computer program]," *San Mateo, CA: Electronic Arts*, 1983.

[43] J. E. Bardram, "The Java Context Awareness Framework (JCAF) - a service infrastructure and programming framework for context-aware applications," *Proceedings of the 3rd International Conference on Pervasive Computing (Pervasive 2005)*, 2005.

[44] A. Schmidt, K. A. Aidoo, A. Takaluoma, U. Tuomela, K. V. Laerhoven, and W. V. de Velde, "Advanced interaction in context," *Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*, pp. 89–101, 1999.

[45] A. Schmidt, M. Beigl, and H. W. Gellersen, "There is more to context than location," *Computers & Graphics*, vol. 23, pp. 893–901, 1999.

[46] H. W. Gellersen, A. Schmidt, and M. Beigl, "Multi-sensor context-awareness in mobile devices and smart artifacts," *Mob. Netw. Appl.*, vol. 7, pp. 341–351, 2002.

[47] M. Raento, A. Oulasvirta, R. Petit, and H. Toivonen, "Contextphone: A prototyping platform for context-aware mobile applications," *IEEE Pervasive Computing*, vol. 4, pp. 51–59, 2005.

[48] A. Oulasvirta, R. Petit, and M. Raento, "Interpreting and acting on mobile awareness cues," *Human-Computer Interaction*, vol. 22, pp. 97–135, 2007.

[49] A. Oulasvirta, M. Raento, and S. Tiitta, "ContextContacts: re-designing smartphone's contact book to support mobile awareness and collaboration," *Proceedings of the 7th international conference on Human computer interaction with mobile devices & services*, pp. 167–174, 2005.

[50] J. E. Bardram and T. R. Hansen, "The AWARE architecture: supporting context-mediated social awareness in mobile cooperation," *Proceedings of the 2004 ACM conference on Computer supported cooperative work*, pp. 192–201, 2004.

[51] K. Nagel, C. D. Kidd, T. O'Connell, A. K. Dey, and G. D. Abowd, "The family intercom: Developing a context-aware audio communication system," *Proceedings of the 3rd international conference on Ubiquitous Computing*, pp. 176–183, 2001.

[52] R. N. Myers and E. Zapata, "Linking information for mobile use," Master's Thesis, Naval Postgraduate School, Sep. 2007, physical description: xiv, 79 p. : ill. (some col.) ; 28 cm.

[53] M. Akbas, "Personal information search on mobile devices," Master's Thesis, Naval Postgraduate School, Sep. 2007, physical description: xii, 89 p. : col. ill. ; 28 cm.

[54] C. Soanes, *Compact Oxford English Dictionary*, 3rd ed. Oxford University Press, Jul. 2005.

[55] G. D. Abowd and E. D. Mynatt, "Charting past, present, and future research in ubiquitous computing," *ACM Trans. Comput.-Hum. Interact.*, vol. 7, no. 1, pp. 29–58, 2000.

[56] G. Abowd, E. Mynatt, and T. Rodden, "The human experience [of ubiquitous computing]," *Pervasive Computing, IEEE*, vol. 1, no. 1, pp. 48– 57, 2002.

[57] Y. Miyata and D. A. Norman, "Psychological issues in support of multiple activities," in *User Centered System Design.* Lawrence Erlbaum Associates, 1986, pp. 265–284.

[58] V. Kaptelinin and B. A. Nardi, "Activity theory in a nutshell," in *Acting with Technology: Activity Theory and Interaction Design.* The MIT Press, Oct. 2006, pp. 29–72.

[59] V. Kaptelinin, "Activity theory: implications for human-computer interaction," in *Context and consciousness: activity theory and human-computer interaction.* Massachusetts Institute of Technology, 1995, pp. 103–116.

[60] B. A. Nardi, "Activity theory and human-computer interaction," in *Context and consciousness: activity theory and human-computer interaction.* Massachusetts Institute of Technology, 1995, pp. 7–16.

[61] K. Kuutti, "Activity theory as a potential framework for human-computer interaction research," in *Context and consciousness: activity theory and human-computer interaction.* Massachusetts Institute of Technology, 1995, pp. 17–44.

[62] L. J. Bannon, "From human factors to human actors: The role of psychology and human-computer interaction studies in system design," *Design at Work: Cooperative Design of Computer Systems*, 1991.

[63] B. Brown and R. Randell, “Building a context sensitive telephone: some hopes and pitfalls for context sensitive computing,” *Computer Supported Cooperative Work (CSCW)*, vol. 13, no. 3, pp. 329–345, 2004.

[64] F. D. Davis, “Perceived usefulness, perceived ease of use, and user acceptance of information technology,” *MIS Quarterly*, vol. 13, no. 3, pp. 319–340, Sep. 1989.

[65] M. Keil, P. M. Beranek, and B. R. Konsynski, “Usefulness and ease of use: field study evidence regarding task considerations,” *Decision Support Systems*, vol. 13, no. 1, pp. 75–91, 1995.

[66] D. H. J. Rubin and D. I. S. Rubin, *Qualitative Interviewing: The Art of Hearing Data*, 2nd ed. Sage Publications, Inc, Aug. 2004.

[67] M. Weiser, “The computer for the 21st century,” *Scientific American*, vol. 265, pp. 94–104, Sep. 1991.

[68] D. Garlan, D. P. Siewiorek, A. Smailagic, and P. Steenkiste, “Project Aura: toward distraction-free pervasive computing,” *Pervasive Computing, IEEE*, vol. 1, pp. 22–31, 2002.

[69] P. J. Denning, “Infoglut,” *Communications of the ACM*, vol. 49, pp. 15–19, 2006.

[70] F. M. Shipman III and C. C. Marshall, “Spatial hypertext: an alternative to navigational and semantic links,” *ACM Comput. Surv*, vol. 31, p. 14, 1999.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvior, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California